Jacob Ciurca

Dr. Lu

Lab 6

Cosc 320

10/14/2020

**Pre-Lab:** review set and map

**Lab:**

**stateCity.h:**

#ifndef STATECITY\_CLASS

#define STATECITY\_CLASS

#include <iostream>

#include <string>

using namespace std;

// object stores the state name and city in the state

class stateCity{

public:

stateCity(string s, string c){

state = s;

city = c;

};

string getState(){

return state; //Defined stateCity in main instead of a separate file

};

string getCity(){

return city;

};

friend ostream& operator<< (ostream &ostr, const stateCity& obj);

friend bool operator< (const stateCity& lhs, const stateCity& rhs);

friend bool operator== (const stateCity& lhs, const stateCity& rhs);

private:

string state;

string city;

};

ostream& operator<< (ostream &ostr, const stateCity& obj){

ostr << "(" << obj.state << ", " << obj.city << ")";

return ostr;

}

bool operator< (const stateCity& lhs, const stateCity& rhs){

if(lhs.city < rhs.city){

return true;

}

else if(lhs.city == rhs.city && lhs.state < rhs.state){ //Operator< used to insert objects into set

return true; //Must return true if either new city or if the city is in a different state

}

else{

return false;

}

}

bool operator== (const stateCity& lhs, const stateCity& rhs){

if(lhs.state == rhs.state){

return true;

}

else{ //Operator== used to create find function later in lab

return false;

}

}

void findState(const set<stateCity>& s, stateCity value){

set<stateCity>::const\_iterator iter = s.begin();

int count = 0;

for(iter = s.begin(); iter != s.end(); ++iter){ //find function, checks state value using == operator overloading and outputs state name if true

if(\*iter == value){

cout << \*iter << endl;

count++;

}

}

if(count == 0){

cout << value.getState() << " not found in set\n";

}

}

#endif // STATECITY\_CLASS

**Lab06\_set.cpp:**

#include <iostream>

#include <set>

using namespace std;

class stateCity{

public:

stateCity(string s, string c){

state = s;

city = c;

};

string getState(){

return state; //Defined stateCity in main instead of a separate file

};

string getCity(){

return city;

};

friend ostream& operator<< (ostream &ostr, const stateCity& obj);

friend bool operator< (const stateCity& lhs, const stateCity& rhs);

friend bool operator== (const stateCity& lhs, const stateCity& rhs);

private:

string state;

string city;

};

ostream& operator<< (ostream &ostr, const stateCity& obj){

ostr << "(" << obj.state << ", " << obj.city << ")";

return ostr;

}

bool operator< (const stateCity& lhs, const stateCity& rhs){

if(lhs.city < rhs.city){

return true;

}

else if(lhs.city == rhs.city && lhs.state < rhs.state){ //Operator< used to insert objects into set

return true; //Must return true if either new city or if the city is in a different state

}

else{

return false;

}

}

bool operator== (const stateCity& lhs, const stateCity& rhs){

if(lhs.state == rhs.state){

return true;

}

else{ //Operator== used to create find function later in lab

return false;

}

}

void findState(const set<stateCity>& s, stateCity value){

set<stateCity>::const\_iterator iter = s.begin();

int count = 0;

for(iter = s.begin(); iter != s.end(); ++iter){ //find function, checks state value using == operator overloading and outputs state name if true

if(\*iter == value){

cout << \*iter << endl;

count++;

}

}

if(count == 0){

cout << value.getState() << " not found in set\n";

}

}

int main()

{

stateCity sc1("Maryland", "Salisbury");

stateCity sc2("Virginia", "Richmond");

stateCity sc3("Maryland", "Baltimore");

stateCity sc4("Florida", "Miami");

stateCity sc5("Texas", "Houston");

set<stateCity> sc;

sc.insert(sc1);

sc.insert(sc2);

sc.insert(sc3);

sc.insert(sc4);

sc.insert(sc5);

cout << "Please enter a state to search for in the set.\n";

string State;

cin >> State;

stateCity sc6(State, ""); //declare state object with empty city to search set

cout << "Searching for stateCity objects with the state " << State << " in the set\n";

findState(sc, sc6);

return 0;

}

**Lab06\_map.cpp:**

#include <iostream>

#include <map>

using namespace std;

int main()

{

map<string, string> m;

m["Maryland"] = "Baltimore";

m["New York"] = "New York City";

m["Virginia"] = "Richmond";

m["Florida"] = "Miami";

m["Texas"] = "Austin";

map<string, string>::const\_iterator iter;

cout << "Please enter a state name to search in the map\n";

string state;

cin >> state;

int count = 0;

for(iter = m.begin(); iter != m.end(); ++iter){

if(iter->first == state){

cout << "(" << iter->first << ", " << iter->second << ")" << endl; //Search through map looking for the state as key

count++; //if found increase count

}

}

if(count == 0){ //if state is not found within map

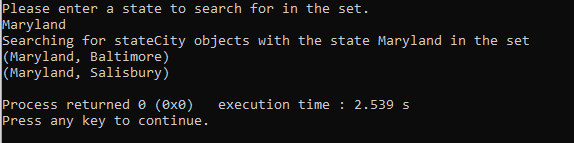
cout << "Map entry with state name " << state << " not found within the map\n";

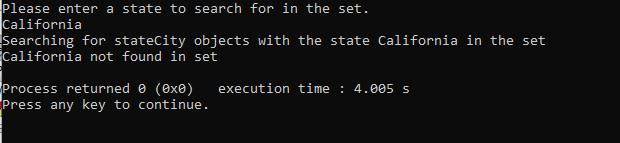
}

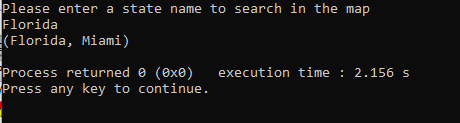
return 0;

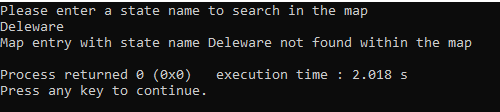
}

**Sample Outputs:**









**Post Lab:** Comparing the two outputs you can see that they do the same general thing with one difference. With a set you can have multiple objects with the same state when you use proper operator overloading. With a map you can only have one entry per state since we used the state name as a key, and you can only have one instance of that key value in the map since it must be unique to be a key. This lab took me about 1 hour to complete and I did it by myself with no help.