Автор: Білий Вадим, КІТ-119а

Дата: 27.05.2020

Лабораторна робота 13. АЛГОРИТМИ ПЕРЕМІЩЕННЯ ТА ПОШУКУ

Тема. STL. Алгоритми переміщення та пошуку.

Мета – на практиці порівняти STL-алгоритми, що не модифікують послідовність.

Загальне завдання

Поширити попередню лабораторну роботу, додаючи такі можливості діалогового меню:

- виведення всіх елементів масиву за допомогою STL-функції for_each;
- визначення кількості елементів за заданим критерієм;
- пошук елемента за заданим критерієм.

Опис класів

Клас ргз: C_Rgz

Клас наслідник ргз: C_RgzM

Опис змінних

string object; - назва об’єкту

int mark; - оцінка

vector<C_Rgz*> vect; - вектор

list <C_Rgz*> lis; - список

map <int,C_Rgz*> mp; - дерево (ключ, данні)

set <C_Rgz*> st; - дерево(ключ)

Опис методів

virtual void setObject(const string str); -сетер

virtual string getObject() const; - гетер

virtual string getString() const; - повертає строку з даними

virtual void input(istream& a); - ввід

friend ostream& operator<< (ostream& output, C_Rgz& obj); - перевантаження <<

virtual bool operator==(C_Rgz& obj); - перевантаження ==

virtual C_Rgz& operator= (C_Rgz& temp); - перевантаження =

friend istream& operator>> (istream& input, C_Rgz& obj); - перевантаження>>

void setMark(const int a); - сетер

int getMark()const; - гетер

virtual void input(istream& a); -ввід

virtual bool operator==(C_RgzM& obj); - перевантаження ==

virtual C_Rgz& operator= (C_RgzM& temp); - перевантаження =

string getString() const override; - повертає строку з даними

Текст програми

C_Rgz.cpp

```
#include "C_Rgz.h"

void C_Rgz::setObject(const string str)
{
	object = str;
}

string C_Rgz::getObject() const
{
	return object;
}

string C_Rgz::getString() const
{
	return object;
}

void C_Rgz::input(istream& a)
{
	a >> object;
}



bool C_Rgz::operator==(C_Rgz& obj)
{
	return getString() == obj.getString();

}

C_Rgz& C_Rgz::operator=(C_Rgz& temp)
{
	object = temp.getObject();
	return *this;
}
```

```
C_Rgz::C_Rgz():object("Nothing")
{

}

C_Rgz::C_Rgz(string str):object(str)
{
}

C_Rgz::C_Rgz(C_Rgz& a):object(a.getObject())
{
}

ostream& operator<<(ostream& output, C_Rgz& obj)
{
	output << obj.getObject();
	return output;
}

istream& operator>>(istream& input, C_Rgz& obj)
{
	obj.input(input);
	return input;
}
```

C_RgzM.cpp

```
#include "C_RgzM.h"

void C_RgzM::setMark(const int a)
{
	mark = a;
}

int C_RgzM::getMark() const
{
	return mark;
}

void C_RgzM::input(istream& a)
{
	a >> object >> mark;
}

bool C_RgzM::operator==(C_RgzM& obj)
{
	return getString()==obj.getString();
}

C_Rgz& C_RgzM::operator=(C_RgzM& temp)
{
	object = temp.getObject();
	mark = temp.getMark();
	return *this;
}

string C_RgzM::getString()const
{
	stringstream ss;
	ss << object << " " << mark;
	return ss.str();
}
```

```
C_RgzM::C_RgzM():mark(0)
{
	setObject("Nothing");
}

C_RgzM::C_RgzM(string str, int m):mark(m)
{
	setObject(str);
}

C_RgzM::C_RgzM(C_RgzM& obj):mark(obj.getMark())
{
	setObject(obj.getObject());
}
```

Source.cpp

```
#include "C_Rgz.h"
#include "C_RgzM.h"
#include <vector>
#include <map>
#include <set>
#include <list>
#include <algorithm>
using std::vector;
using std::map;
using std::list;
using std::set;
using std::endl;
using std::make_pair;
using std::find_if;
void menu();
C_Rgz* tempclass;
C_Rgz* creatEl();
int main() {

	menu();
	if (_CrtDumpMemoryLeaks())
		cout << "\nMemory leack deteckted\n";
	else
		cout << "\nMemory is not leack deteckted\n";

}
void menu() {
	short int choose = 1;
	vector<C_Rgz*> vect;
	vector<C_Rgz*>::iterator itv;
	int count=0;
	while (choose != -1) {
		cout << "\n1-add\n2-delete\n3-show element\n4-show all\n5-find el\n6-count
elem\n-1-exit\nchoose way: ";
		cin >> choose;
		switch (choose)
		{
		case 1:
			vect.push_back(creatEl());
			break;
		case 2:
			cout << "num: ";
			cin >> choose;
			if (!vect.empty()&& 0 < choose<vect.size()) {
				itv = vect.begin();
				delete* (itv + choose);
				vect.erase(itv + choose);
```

```
                }
                break;
        case 3:
                cout << "num: ";
                cin >> choose;
                if (!vect.empty() && 0 < choose < vect.size()) {
                        cout<<vect.at(choose)->getString();
                }

                break;
        case 4:
                for  (C_Rgz* var : vect)
                {
                        cout << var->getString()<<endl;
                }
                break;
        case 5:
                tempclass=creatEl();
                if (find_if(vect.begin(), vect.end(), [](C_Rgz* a) {

                        return a->getString() == tempclass->getString();
                        }) != vect.end()) {
                        cout << "element fount" << endl;
                }
                else {
                        cout << "element not found " << endl;
                        }
                delete tempclass;
                break;
        case 6:
                tempclass = creatEl();
                for (C_Rgz* var : vect) {
                        if (var->getString() == tempclass->getString()) {
                                count++;
                        }
                }
                cout << "count: "<<count << endl;
                count = 0;
                delete tempclass;
        default:
                break;
        }

    }
    for (C_Rgz* var : vect)
    {
            delete var;
    }
    choose = 0;
    list <C_Rgz*> lis;
    list<C_Rgz*>::iterator itl;
    while (choose != -1) {
            cout << "\n1-add\n2-delete\n3-show element\n4-show all\n5-find el\n6-count
elem\n-1-exit\nchoose way: ";
            cin >> choose;
            switch (choose)
            {
            case 1:
                    lis.push_back(creatEl());
                    break;
            case 2:
                    cout << "num: ";
                    cin >> choose;
                    if (!lis.empty() && 0<choose < lis.size()) {
                            itl = lis.begin();
```

```
                        for (int i = 0; i < choose; i++) {
                                itl++;
                        }
                        delete* itl;
                        lis.erase(itl);
                }
                break;
        case 3:
                cout << "num: ";
                cin >> choose;
                if (!lis.empty() && 0 < choose < lis.size()) {
                        itl = lis.begin();
                        for (int i = 0; i < choose; i++) {
                                itl++;
                        }
                        cout << (*itl)->getString() << endl;
                }
                break;
        case 4:
                for (C_Rgz* var : lis)
                {
                        cout << var->getString() << endl;
                }
        case 5:
                tempclass = creatEl();
                if (find_if(lis.begin(), lis.end(), [](C_Rgz* a) {

                        return a->getString() == tempclass->getString();
                        }) != lis.end()) {
                        cout << "element fount" << endl;
                }
                else {
                        cout << "element not found " << endl;
                }
                delete tempclass;
                break;
        case 6:
                tempclass = creatEl();
                for (C_Rgz* var : lis) {
                        if (var->getString() == tempclass->getString()) {
                                count++;
                        }
                }
                cout << "count: " << count << endl;
                count = 0;
                delete tempclass;
                break;
        default:
                break;
        }
    }
    for (C_Rgz* var : lis)
    {
            delete var;
    }
    bool prz=1;
    choose = 0;
    int key=0;
    vector<int> keymass;
    map <int,C_Rgz*> mp;
    map<int, C_Rgz*>::iterator itm;
    while (choose != -1) {
            cout << "\n1-add\n2-delete\n3-show element\n4-show all\n5-find el\n6-count
elem\n-1-exit\nchoose way: ";
            cin >> choose;
```

```
		switch (choose)
		{
		case 1:
			cout << "\nkey: ";
			cin >> key;
			for (int i = 0; i < keymass.size(); i++) {
				if (key == keymass[i]) {
					prz = 0;
				}
			}
			if (prz) {
				mp.insert(make_pair(key, creatEl()));
				keymass.push_back(key);
			}
			prz = 1;
			break;
		case 2:
			cout << "\nkey: ";
			cin >> key;
			itm = mp.find(key);
			if (itm != mp.end()) {
				delete itm->second;

				mp.erase(key);
				for (int i = 0; i < keymass.size(); i++) {
					if (key == keymass[i]) {
						keymass.erase(keymass.begin()+i);
					}
				}
			}
			break;
		case 3:
			cout << "key: ";
			cin >> key;
			itm = mp.find(key);
			if (itm != mp.end())
				cout << itm->first<<" "<<itm->second->getString();
			break;
		case 4:
			for (int var : keymass) {
				itm = mp.find(var);
				if (itm != mp.end())
					cout << itm->first << " " << itm->second-
>getString()<<endl;
			}
			break;

		case 5:
			tempclass = creatEl();
			if (find_if(mp.begin(), mp.end(), [](auto a) {

				return a.second->getString() == tempclass->getString();
				}) != mp.end()) {
				cout << "element fount" << endl;
			}
			else {
				cout << "element not found " << endl;
			}
			delete tempclass;
			break;
		case 6:
			tempclass = creatEl();
			for (auto var : mp) {
				if (var.second->getString() == tempclass->getString()) {
					count++;
```

```
                        }
                    }
                    cout << "count: " << count << endl;
                    count = 0;
                    delete tempclass;
                    break;
                default:
                    break;
                }
        }
        for (int var : keymass) {
                itm = mp.find(var);
                if (itm != mp.end())
                    delete itm->second;
        }
        prz = 0;
        int is=0;
        choose = 0;
        C_Rgz* el;
        // key = 0;
        //vector<C_Rgz*> keymass;
        set <C_Rgz*> st;
        set <C_Rgz*>::iterator its;
        while (choose != -1) {
                cout << "\n1-add\n2-delete\n3-show element\n4-show all\n5-find el\n6-count
elem\n-1-exit\nchoose way: ";
                cin >> choose;
                switch (choose)
                {
                case 1:
                    st.insert(creatEl());
                    break;
                case 2:
                    el = creatEl();
                    its = st.begin();
                    for (C_Rgz* var : st)
                        if (var->getString() == el->getString()&&!prz) {
                            prz = 1;
                            for (int i = 0; i < is; i++)
                                its++;
                            break;
                        }
                        else {
                            if(!prz)
                            is++;
                        }
                    if (prz) {
                        delete* its;
                        st.erase(its);
                    }
                    delete el;
                    break;
                case 3:
                    el = creatEl();
                    for (C_Rgz* var : st)

                        if (el->getString() == var->getString()) {
                            cout << var->getString()<<endl;
                        }
                    delete el;

                    break;
                case 4:
                    for (C_Rgz* var : st)
                        cout << var->getString() << endl;
```

```
                        break;
                case 5:
                        tempclass = creatEl();
                        if (find_if(st.begin(), st.end(), [](auto a) {

                                return a->getString() == tempclass->getString();
                                }) != st.end()) {
                                cout << "element fount" << endl;
                        }
                        else {
                                cout << "element not found " << endl;
                        }
                        delete tempclass;
                        break;
                case 6:
                        tempclass = creatEl();
                        for (auto var : st) {
                                if (var->getString() == tempclass->getString()) {
                                        count++;
                                }
                        }
                        cout << "count: " << count << endl;
                        count = 0;
                        delete tempclass;
                        break;
                default:
                        break;
                }
        }
        for (C_Rgz* var : st)
                delete var;
}
C_Rgz* creatEl()
{
        int choose;
        cout << "\n1-Rgz\n2-RgzM\nchoose: ";
        cin >> choose;
        cout << endl;
        C_Rgz* a;
        if (choose == 1) {
                a = new C_Rgz;
                cout << "\nObject:";
                cin >> *a;
        }
        else
        {
                a = new C_RgzM;
                cout << "\nObject, mark: ";
                cin >> *a;
        }
        return a;
};
```

Test.cpp

```
#include "C_Rgz.h"
#include "C_RgzM.h"
#include <vector>
#include <map>
#include <set>
#include <list>
using std::vector;
using std::map;
using std::list;
using std::set;
```

```
using std::endl;
using std::make_pair;
short int choose = 1;
int main() {
	vector<C_Rgz*> vect;
	vector<C_Rgz*>::iterator itv;
	vect.push_back(new C_Rgz);

	if (vect.size() == 1)
		cout << "test 1: true" << endl;
	else
		cout << "test 1: false" << endl;
	itv = vect.begin();
	delete* (itv);
	vect.erase(itv);
	if (vect.size() == 0)
		cout << "test 2: true" << endl;
	else
		cout << "test 2: false" << endl;

	for (C_Rgz* var : vect)
	{
		delete var;
	}
	list <C_Rgz*> lis;
	list<C_Rgz*>::iterator itl;
	lis.push_back(new C_Rgz);

	if (lis.size() == 1)
		cout << "test 3: true" << endl;
	else
		cout << "test 3: false" << endl;
	itl = lis.begin();
	delete* itl;
	lis.erase(itl);

	if (lis.size() == 0)
		cout << "test 4: true" << endl;
	else
		cout << "test 4: false" << endl;

	for (C_Rgz* var : lis)
	{
		delete var;
	}
	map <int, C_Rgz*> mp;
	map<int, C_Rgz*>::iterator itm;
	mp.insert(make_pair(50, new C_Rgz));

	if (mp.size() == 1)
		cout << "test 5: true" << endl;
	else
		cout << "test 5: false" << endl;
	itm = mp.find(50);
	delete itm->second;
	mp.erase(50);

	if (mp.size() == 0)
		cout << "test 6: true" << endl;
	else
		cout << "test 6: false" << endl;


	set <C_Rgz*> st;
	set <C_Rgz*>::iterator its;
```

```
	st.insert(new C_Rgz);

	if (st.size() == 1)
		cout << "test 7: true" << endl;
	else
		cout << "test 7: false" << endl;
	its = st.begin();
	delete* its;
	st.erase(its);

	if (st.size() == 0)
		cout << "test 8: true" << endl;
	else
		cout << "test 8: false" << endl;

};
```

C_Rgz.h

```
#pragma once
#include <iostream>
#include <sstream>
using std::string;
using std::istream;
using std::ostream;
using std::cout;
using std::cin;
using std::stringstream;

class C_Rgz
{
protected:
	string object;
public:
	virtual void setObject(const string str);
	virtual string getObject() const;
	virtual string getString() const;
	virtual void input(istream& a);

	 friend ostream& operator<< (ostream& output, C_Rgz& obj);

	virtual bool operator==(C_Rgz& obj);

	virtual C_Rgz& operator= (C_Rgz& temp);

	friend istream& operator>> (istream& input, C_Rgz& obj);


	C_Rgz();
	C_Rgz(string str);
	C_Rgz(C_Rgz &a);
	virtual ~C_Rgz() = default;
};
```

C_RgzM.h

```
#pragma once
#include "C_Rgz.h"
class C_RgzM :
	public C_Rgz
{
private:
```

```
        int mark;
public:
        void setMark(const int a);
        int getMark()const;
        virtual void input(istream& a);


        virtual bool operator==(C_RgzM& obj);

        virtual C_Rgz& operator= (C_RgzM& temp);

        string getString() const override;
        C_RgzM();
        C_RgzM(string str, int m);
        C_RgzM(C_RgzM& obj);
};
```

Висновок

При виконанні даної лабораторної роботи було набуто практичного досвіду роботи з алгоритмами пошуку та переміщення.

Було розроблено програму, що працює з алгоритмами пошуку та переміщення.

Алгоритм пошуку вже реалізований в бібліотеці STL під назвами find, find_if, find_if_not. Так само реалізований алгоритм переміщення під назвою for_each.

Програма протестована, витоків пам’яті немає, виконується без помилок.