Автор: Білий Вадим, КІТ-119а

Дата: 27.05.2020

Лабораторна робота 12. STL

Тема. STL. Ітератори. Послідовні контейнери. Цикл range-for. Асоціативні контейнери.

Мета – отримати базові знання про STL-контейнери. Освоїти основні механізми роботи з STL контейнерами.

Загальне завдання

Маючи класи з прикладної області РГЗ (тільки базовий клас та клас / класи спадкоємці), створити діалогове меню, що дозволяє продемонструвати роботу STL-контейнерів (додавання / видалення / отримання даних, показ усіх елементів) та показати їх принципову різницю:

- vector;

- set;

- list;

- map .

При цьому врахувати, що контейнери містять елементи одного типу, наприклад, базового. Прохід по всьому контейнеру повинен виконуватися за допомогою циклу мови С++11 – range-for.

Додаткове завдання на оцінку «відмінно»:

- контейнери повинні оперувати даними не тільки базового класу, а ще даними класів-спадкоємців.

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>

using std::vector;

using std::map;

using std::list;

using std::set;

using std::endl;

using std::make\_pair;

void menu();

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;

while (choose != -1) {

cout << "\n1-add\n2-delete\n3-show element\n4-show all\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;

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\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;

}

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\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;

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\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;

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 STL контейнерами.

Було розроблено програму, що працює з такими STL контейнерами: vector, list, map, set.

vector – вектор.

list – список.

map – дерево (ключ, данні).

set – дерево(ключ).

STL – це бібліотека шаблонних класів, шо містять стандартні методи збереження пм’яті.

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