Московский Авиационный Институт

(Национальный Исследовательский Университет)

Кафедра 806 «Вычислительная информатика и программирование»

Факультет: «Информационные технологии и прикладная математика»

Лабораторная работа

Дисциплина: «Объектно-ориентированное программирование»

III семестр

Задание 5: «Основы работы с коллекциями: итераторы»

|  |  |
| --- | --- |
| Группа: | М8О-208Б-18, №3 |
| Студент: | Анисимов Валерий Алексеевич |
| Преподаватель: | Журавлёв Андрей Андреевич |
| Оценка: |  |
| Дата: | 25.11.2019 |

Москва, 2019

1. **Тема**: Основы работы с коллекциями: итераторы
2. **Цель работы**: Изучение основ работы с коллекциями, знакомство с шаблоном проектирования «Итератор»
3. **Задание** (*вариант № 3* ):

Фигура — прямоугольник. Контейнер — стек.

1. **Адрес репозитория на GitHub** [https://github.com/wAlienUFOx/oop\_exercise\_0](https://github.com/wAlienUFOx/oop_exercise_01)5
2. **Код программы на С++**

main.cpp

#include <iostream>

#include <algorithm>

#include "rectangle.h"

#include "containers/stack.h"

int main() {

size\_t N;

float S;

char option = '0';

containers::stack<Rectangle<int>> st;

Rectangle<int> \*rec = nullptr;

while (option != 'q') {

std::cout << "choose option (m - man)" << std::endl;

std::cin >> option;

switch (option) {

case 'q':

break;

case 'm':

std::cout << "1) push new element into stack\n"

<< "2) insert element into chosen position\n"

<< "3) pop element from the stack\n"

<< "4) delete element from the chosen position\n"

<< "5) print stack\n"

<< "6) count elements with area less then chosen value\n"

<< "7) print top element\n"

<< "q) - quit" << std::endl;

break;

case '1': {

try{

rec = new Rectangle<int>(std::cin);

}catch(std::logic\_error& err){

std::cout << err.what() << std::endl;

break;

}

st.push(\*rec);

break;

}

case '2': {

std::cout << "enter position to insert" << std::endl;

std::cin >> N;

std::cout << "enter rectangle" << std::endl;

try{

rec = new Rectangle<int>(std::cin);

}catch(std::logic\_error& err){

std::cout << err.what() << std::endl;

break;

}

st.insert\_by\_number(N, \*rec);

break;

}

case '3': {

st.pop();

break;

}

case '4': {

std::cout << "enter position to delete" << std::endl;

std::cin >> N;

st.delete\_by\_number(N);

break;

}

case '5': {

std::for\_each(st.begin(), st.end(), [](Rectangle<int> &REC) { REC.Print(std::cout); });

break;

}

case '6': {

std::cout << "enter max area" <<std::endl;

std::cin >> S;

std::cout << std::count\_if(st.begin(), st.end(), [=](Rectangle<int> &X) { return X.Area() < S; })

<< std::endl;

break;

}

case '7' : {

st.top().Print(std::cout);

break;

}

default:

std::cout << "Wrong. Try m - manual" << std::endl;

break;

}

}

return 0;

}

point.h

#ifndef OOP\_LAB5\_POINT\_H

#define OOP\_LAB5\_POINT\_H

#include <iostream

template<class T>

struct point {

T x;

T y;

};

template<class T>

std::istream& operator>>(std::istream& is, point<T>& p) {

is >> p.x >> p.y;

return is;

}

template<class T>

std::ostream& operator<<(std::ostream& os, point<T> p) {

os << '(' << p.x << ' ' << p.y << ')';

return os;

}

#endif

rectangle.cpp

#ifndef OOP\_LAB5\_RECTANGLE\_H

#define OOP\_LAB5\_RECTANGLE\_H

#include "point.h"

#include <cmath>

template <class T>

class Rectangle {

public:

point<T> A , B, C, D;

explicit Rectangle<T>(std::istream& is) {

is >> A >> B >> C >> D;

double a, b, c, d, d1, d2, ABC, BCD, CDA, DAB;

a = sqrt((B.x- A.x) \* (B.x - A.x) + (B.y - A.y) \* (B.y - A.y));

b = sqrt((C.x- B.x) \* (C.x - B.x) + (C.y - B.y) \* (C.y - B.y));

c = sqrt((C.x- D.x) \* (C.x - D.x) + (C.y - D.y) \* (C.y - D.y));

d = sqrt((D.x- A.x) \* (D.x - A.x) + (D.y - A.y) \* (D.y - A.y));

d1 = sqrt((B.x- D.x) \* (B.x - D.x) + (B.y - D.y) \* (B.y - D.y));

d2 = sqrt((C.x- A.x) \* (C.x - A.x) + (C.y - A.y) \* (C.y - A.y));

ABC = (a \* a + b \* b - d2 \* d2) / 2 \* a \* b;

BCD = (b \* b + c \* c - d1 \* d1) / 2 \* b \* c;

CDA = (d \* d + c \* c - d2 \* d2) / 2 \* d \* c;

DAB = (a \* a + d \* d - d1 \* d1) / 2 \* a \* d;

if(ABC != BCD || ABC != CDA || ABC != DAB)

throw std::logic\_error("It`s not a rectangle");

}

Rectangle<T>() = default;

double Area() {

double a = sqrt((B.x - A.x) \* (B.x - A.x) + (B.y - A.y) \* (B.y - A.y));

double b = sqrt((C.x - B.x) \* (C.x - B.x) + (C.y - B.y) \* (C.y - B.y));

return a \* b;

}

void Print(std::ostream& os) {

os << A << " " << B << " " << C << " " << D << std::endl;

}

void operator<< (std::ostream& os) {

os << A << " " << B << " " << C << " " << D;

}

};

#endif

stack.h

#ifndef OOP\_EXERCISE\_05\_STACK\_H

#define OOP\_EXERCISE\_05\_STACK\_H

#include <iterator>

#include <memory>

#include <algorithm>

namespace containers {

template<class T>

class stack {

private:

struct element;

size\_t size = 0;

public:

stack() = default;

class forward\_iterator {

public:

using value\_type = T;

using reference = T&;

using pointer = T\*;

using difference\_type = std::ptrdiff\_t;

using iterator\_category = std::forward\_iterator\_tag;

explicit forward\_iterator(element\* ptr);

T& operator\*();

forward\_iterator& operator++();

forward\_iterator operator++(int);

bool operator== (const forward\_iterator& other) const;

bool operator!= (const forward\_iterator& other) const;

private:

element\* it\_ptr;

friend stack;

};

forward\_iterator begin();

forward\_iterator end();

void push(const T& value);

T& top();

void pop();

void delete\_by\_it(forward\_iterator d\_it);

void delete\_by\_number(size\_t N);

void insert\_by\_it(forward\_iterator ins\_it, T& value);

void insert\_by\_number(size\_t N, T& value);

private:

struct element {

T value;

std::unique\_ptr<element> next\_element = nullptr;

forward\_iterator next();

};

std::unique\_ptr<element> first = nullptr;

};

template<class T>

typename stack<T>::forward\_iterator stack<T>::begin() {

return forward\_iterator(first.get());

}

template<class T>

typename stack<T>::forward\_iterator stack<T>::end() {

return forward\_iterator(nullptr);

}

template<class T>

void stack<T>::push(const T& value) {

if (first == nullptr){

first = std::unique\_ptr<element>(new element{value});

}else{

auto \*tmp = new element{value};

std::swap(tmp->next\_element, first);

first = std::move(std::unique\_ptr<element>(tmp));

}

size++;

}

template<class T>

void stack<T>::pop() {

if (size == 0) {

throw std::logic\_error ("stack is empty");

}

first = std::move(first->next\_element);

size--;

}

template<class T>

T& stack<T>::top() {

if (size == 0) {

throw std::logic\_error ("stack is empty");

}

return first->value;

}

template<class T>

void stack<T>::delete\_by\_it(containers::stack<T>::forward\_iterator d\_it) {

forward\_iterator i = this->begin(), end = this->end();

if (d\_it == end) throw std::logic\_error ("out of borders");

if (d\_it == this->begin()) {

this->pop();

return;

}

while((i.it\_ptr != nullptr) && (i.it\_ptr->next() != d\_it)) {

++i;

}

if (i.it\_ptr == nullptr) throw std::logic\_error ("out of borders");

i.it\_ptr->next\_element = std::move(d\_it.it\_ptr->next\_element);

size--;

}

template<class T>

void stack<T>::delete\_by\_number(size\_t N) {

forward\_iterator it = this->begin();

for (size\_t i = 1; i <= N; ++i) {

if (i == N) break;

++it;

}

this->delete\_by\_it(it);

}

template<class T>

void stack<T>::insert\_by\_it(containers::stack<T>::forward\_iterator ins\_it, T& value) {

auto tmp = std::unique\_ptr<element>(new element{value});

forward\_iterator i = this->begin();

if (ins\_it == this->begin()) {

tmp->next\_element = std::move(first);

first = std::move(tmp);

size++;

return;

}

while((i.it\_ptr != nullptr) && (i.it\_ptr->next() != ins\_it)) {

++i;

}

if (i.it\_ptr == nullptr) throw std::logic\_error ("out of borders");

tmp->next\_element = std::move(i.it\_ptr->next\_element);

i.it\_ptr->next\_element = std::move(tmp);

size++;

}

template<class T>

void stack<T>::insert\_by\_number(size\_t N, T& value) {

forward\_iterator it = this->begin();

for (size\_t i = 1; i <= N; ++i) {

if (i == N) break;

++it;

}

this->insert\_by\_it(it, value);

}

template<class T>

typename stack<T>::forward\_iterator stack<T>::element::next() {

return forward\_iterator(this->next\_element.get());

}

template<class T>

stack<T>::forward\_iterator::forward\_iterator(containers::stack<T>::element \*ptr) {

it\_ptr = ptr;

}

template<class T>

T& stack<T>::forward\_iterator::operator\*() {

return this->it\_ptr->value;

}

template<class T>

typename stack<T>::forward\_iterator& stack<T>::forward\_iterator::operator++() {

if (it\_ptr == nullptr) throw std::logic\_error ("out of stack borders");

\*this = it\_ptr->next();

return \*this;

}

template<class T>

typename stack<T>::forward\_iterator stack<T>::forward\_iterator::operator++(int) {

forward\_iterator old = \*this;

++\*this;

return old;

}

template<class T>

bool stack<T>::forward\_iterator::operator==(const forward\_iterator& other) const {

return it\_ptr == other.it\_ptr;

}

template<class T>

bool stack<T>::forward\_iterator::operator!=(const forward\_iterator& other) const {

return it\_ptr != other.it\_ptr;

}

}

#endif

CMakeLists.txt

cmake\_minimum\_required (VERSION 3.5)

project(lab5)

add\_executable(oop\_exercise\_05

main.cpp)

set(CMAKE\_CXX\_FLAGS "${CMAKE\_CXX\_FLAGS} -Wall -Wextra")

set\_target\_properties(oop\_exercise\_05 PROPERTIES CXX\_STANDART 14 CXX\_STANDART\_REQUIRED ON)

1. **Набор testcases**

test\_01.txt Ожидаемое действие

1 push (0,0)(0,1)(1,1)(1,0)

0 0 0 1 1 1 1 0

1 push (1,1)(1,3)(3,3)(3,1)

1 1 1 3 3 3 3 1

5 Печать стека

3 pop

5 Печать стека

q Выход

test\_02.txt Ожидаемое действие

1 Не является прямоугольником

0 0 1 1 2 1 2 0

1 push (-2,2)(-2,4)(4,4)(4,2)

-2 2 -2 4 4 4 4 2

2 Вставка (0,0)(0,1)(1,1)(1,0) на

1 позицию 1

0 0 0 1 1 1 1 0

5 Печать стека

6 Вывод количества элементов,

2 площадь которых < 2 (1)

7

q Выход

test\_03.txt Ожидаемое действие

2 Вставка (0,0)(1,1)(2,0)(-1,1) на

1 позицию 1

0 0 1 1 2 0 1 -1

5 Печать стека

4 Удаление элемента с

1 позиции 1

5 Печать стека

q Выход

1. **Результаты выполнения тестов**

walien@PC-name:~/2kurs/CPP/lab5/tmp$ ./oop\_exercise\_05 < ~/2kurs/CPP/lab5/test\_01.txt

choose option (m - man)

choose option (m - man)

choose option (m - man)

(1 1) (1 3) (3 3) (3 1)

(0 0) (0 1) (1 1) (1 0)

choose option (m - man)

choose option (m - man)

(0 0) (0 1) (1 1) (1 0)

choose option (m - man)

walien@PC-name:~/2kurs/CPP/lab5/tmp$ ./oop\_exercise\_05 < ~/2kurs/CPP/lab5/test\_02.txt

choose option (m - man)

It`s not a rectangle

choose option (m - man)

choose option (m - man)

enter position to insert

enter rectangle

choose option (m - man)

(0 0) (0 1) (1 1) (1 0)

(-2 2) (-2 4) (4 4) (4 2)

choose option (m - man)

enter max area

1

choose option (m - man)

(0 0) (0 1) (1 1) (1 0)

choose option (m - man)

walien@PC-name:~/2kurs/CPP/lab5/tmp$ ./oop\_exercise\_05 < ~/2kurs/CPP/lab5/test\_03.txt

choose option (m - man)

enter position to insert

enter rectangle

choose option (m - man)

(0 0) (1 1) (2 0) (1 -1)

choose option (m - man)

enter position to delete

choose option (m - man)

choose option (m - man)

1. **Объяснение результатов работы программы - вывод**

Методы и члены коллекции:  
size — размер коллекции  
element — описание элемента коллекции  
first — головной элемент коллекции  
push — добавление элемнета в стек  
pop — удаление элекмента из стека  
top — возвращает значение головного элемента стека  
delete\_by\_it — удаление элемента по итератору  
delete\_by\_number — уделение элемента по номеру  
insert\_by\_it — вставка элемента по итератору  
insert\_by\_number — удаление элемента по итератору  
forward\_iterator — реализация итератора типа forward\_iterator

В ходе данной лабораторной работы были получены навыки работы с умными указателями, в частности unique\_ptr, а так же навыки написания итераторов, совместимыми со стандартными функциями (std::for\_each, std::count\_if).

Умные указатели полезны в работе с динамечскими структурами, как инструменты более удобного контроля за выделением и освобождением ресурсов, что помогает избежать утечек памяти.