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, №2 |
| Студент: | Алексеева Мария Алексеевна |
| 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 № 2* ):

Фигура — квадрат. Контейнер — стек.

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

main.cpp

#include <iostream>

#include <algorithm>

#include "square.h"

#include "conteiners/stack.h"

int main() {

size\_t N;

float S;

char option = '0';

containers::stack<Square<int>> q;

Square<int> kva{};

while (option != 'q') {

std::cout << "выберите опцию (m for man, q to quit)" << std::endl;

std:: cin >> option;

switch (option) {

case 'q':

break;

case 'm':

std::cout << "1) добавить новый элемент в стэк\n"

<< "2) вставить элемент на позицию\n"

<< "3) (pop)удаление верхнего элемента\n"

<< "4) (top) значение вернего элемента\n"

<< "5) удалить элемент с позиции\n"

<< "6) напечатать стэк\n"

<< "7) количесто элементов с площадью меньше чем \n" << std::endl;

break;

case '1': {

std::cout << "введите вершины квадрата: " << std::endl;

kva = Square<int>(std::cin);

try{

kva.Check();

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

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

break;

}

q.push(kva);

break;

}

case '2': {

std::cout << "позиция для вставки: ";

std::cin >> N;

std::cout << "введите квадрат: ";

kva =Square<int>(std::cin);

q.insert\_by\_number(N, kva);

break;

}

case '3': {

q.pop();

break;

}

case '4': {

q.top().Printout(std::cout);

break;

}

case '5': {

std::cout << "позиция для удаления: ";

std::cin >> N;

q.delete\_by\_number(N);

break;

}

case '6': {

std::for\_each(q.begin(), q.end(), [](Square<int> &X) { X.Printout(std::cout); });

break;

}

case '7': {

std::cout << "площадь для сравнения: ";

std::cin >> S;

std::cout <<"количестов элементов с площадью меньше чем" << S << " :" << std::count\_if(q.begin(), q.end(), [=](Square<int>& X){return X.Area() < S;}) << std::endl;

break;

}

default:

std::cout << "нет такой опции. Попробуйте m" << std::endl;

break;

}

}

return 0;

}

square.h

#ifndef SQUARE

#define SQUARE

#include "vertex.h"

template <class T>

class Square {

public:

vertex<T> points[4];

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

for (auto & point : points) {

is >> point;

}

}

Square<T>() = default;

double Area() {

double res = 0;

for (size\_t i = 0; i < 3; i++) {

res += (points[i].x \* points[i+1].y) - (points[i+1].x \* points[i].y);

}

res = res + (points[3].x \* points[0].y) - (points[0].x \* points[3].y);

return std::abs(res)/ 2;

}

void Printout(std::ostream& os) {

for (int i = 0; i < 4; ++i) {

os << this->points[i];

if (i != 3) {

os << ", ";

}

}

os << std::endl;

}

void Check() {

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

a = sqrt((points[2].x- points[1].x) \* (points[2].x - points[1].x) + (points[2].y - points[1].y) \* (points[2].y - points[1].y));

b = sqrt((points[3].x- points[2].x) \* (points[3].x - points[2].x) + (points[3].y - points[2].y) \* (points[3].y - points[2].y));

c = sqrt((points[3].x- points[4].x) \* (points[3].x - points[4].x) + (points[3].y - points[4].y) \* (points[3].y - points[4].y));

d = sqrt((points[4].x- points[1].x) \* (points[4].x - points[1].x) + (points[4].y - points[1].y) \* (points[4].y - points[1].y));

d1 = sqrt((points[2].x- points[4].x) \* (points[2].x - points[4].x) + (points[2].y - points[4].y) \* (points[2].y - points[4].y));

d2 = sqrt((points[3].x- points[1].x) \* (points[3].x - points[1].x) + (points[3].y - points[1].y) \* (points[3].y - points[1].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 || a!=b || a!=c || a!=d )

throw std::logic\_error("Это не квадрат!");

}

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

for (int i = 0; i < 4; ++i) {

os << this->points[i];

if (i != 3) {

os << ", ";

}

}

}

};

#endif

vertex.h

#ifndef VERTEX\_H\_

#define VERTEX\_H\_

#include <iostream>

#include <cmath>

template<class T>

struct vertex {

T x;

T y;

};

template<class T>

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

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

return is;

}

template<class T>

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

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

return os;

}

#endif //VERTEX\_H

stack.h

#ifndef STACK\_H

#define STACK\_H

#include <iterator>

#include <memory>

namespace containers {

template<class T>

class stack {

private:

struct element;

int 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; //для арифметики указателей и индексации массива

using iterator\_category = std::forward\_iterator\_tag;//пустой класс для идентификации прямого итератора

explicit forward\_iterator(element\* ptr);

T& operator\*();

forward\_iterator& operator++();

forward\_iterator operator++(int) const;

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();

size\_t length();

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;

};//===============================end-of-class-stack======================================//

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);

}

//=========================base-methods-of-stack==========================================//

template<class T>

size\_t stack<T>::length() {

return size;

}

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;

}

//=================================advanced-methods========================================//

template<class T>

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

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--;

}

//удаление по номеру

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);

}

//==============================iterator`s-stuff=======================================//

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) const {

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 //STACK\_H

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)(1,4)(4,4)(4,1)

1 1 1 4 4 4 4 1

6 Печать стека

3 pop

6 Печать стека

q Выход

test\_02.txt Ожидаемое действие

1 Не является квадратом

0 0 0 1 1 2 1 0

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

-2 2 -2 4 0 4 0 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**

masha@masha-VirtualBox:~/2kurs/oop\_exercise\_05/tmp$ ./oop\_exercise\_05 < ~/2kurs/oop\_exercise\_05/test\_01.txt

выберите опцию (m for man, q to quit)

введите вершины квадрата:

выберите опцию (m for man, q to quit)

введите вершины квадрата:

выберите опцию (m for man, q to quit)

(1 1), (1 4), (4 4), (4 1)

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

выберите опцию (m for man, q to quit)

выберите опцию (m for man, q to quit)

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

выберите опцию (m for man, q to quit)

masha@masha-VirtualBox:~/2kurs/oop\_exercise\_05/tmp$ ./oop\_exercise\_05 < ~/2kurs/oop\_exercise\_05/test\_02.txt

выберите опцию (m for man, q to quit)

введите вершины квадрата:

Это не квадрат!

выберите опцию (m for man, q to quit)

введите вершины квадрата:

выберите опцию (m for man, q to quit)

позиция для вставки: введите квадрат: выберите опцию (m for man, q to quit)

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

(-2 2), (-2 4), (0 4), (0 2)

выберите опцию (m for man, q to quit)

площадь для сравнения: количестов элементов с площадью меньше чем2 :1

выберите опцию (m for man, q to quit)

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

выберите опцию (m for man, q to quit)

masha@masha-VirtualBox:~/2kurs/oop\_exercise\_05/tmp$ ./oop\_exercise\_05 < ~/2kurs/oop\_exercise\_05/test\_03.txt

выберите опцию (m for man, q to quit)

позиция для вставки: введите квадрат: выберите опцию (m for man, q to quit)

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

выберите опцию (m for man, q to quit)

позиция для удаления: выберите опцию (m for man, q to quit)

выберите опцию (m for man, q to quit)

masha@masha-VirtualBox:~/2kurs/oop\_exercise\_05/tmp$ ./oop\_exercise\_05

выберите опцию (m for man, q to quit)

m

1) добавить новый элемент в стэк

2) вставить элемент на позицию

3) (pop)удаление верхнего элемента

4) (top) значение вернего элемента

5) удалить элемент с позиции

6) напечатать стэк

7) количесто элементов с площадью меньше чем

выберите опцию (m for man, q to quit)

2

позиция для вставки: 1

введите квадрат: 0 0 1 1 2 0 1 -1

выберите опцию (m for man, q to quit)

6

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

выберите опцию (m for man, q to quit)

5

позиция для удаления: 1

выберите опцию (m for man, q to quit)

6

выберите опцию (m for man, q to quit)

masha@masha-VirtualBox:~/2kurs/oop\_exercise\_05/tmp$

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

Методы и члены коллекции:  
size — размер коллекции  
element — описание элемента коллекции  
first — головной элемент коллекции  
push — добавление элемента в стек  
pop — удаление элемента из стека  
top — возвращает значение головного элемента стека  
delete\_by\_it — удаление элемента по итератору  
delete\_by\_number — удаление 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 динамическими структурами, как инструменты более удобного контроля за выделением и освобождением ресурсов, что помогает избежать утечек памяти и висячих ссылок.