

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


```

        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 << "количеств 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_STANDARD 14
CXX_STANDARD_REQUIRED ON)

6. Набор testcases

test_01.txt

1
0 0 0 1 1 1 1 0
1
1 1 1 4 4 4 4 1
6
3
6
q

Ожидаемое действие
push (0,0)(0,1)(1,1)(1,0)

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

Печать стека
pop
Печать стека
Выход

test_02.txt

1
0 0 0 1 1 2 1 0
1
-2 2 -2 4 0 4 0 2
2
1
0 0 0 1 1 1 1 0
5
6
2
7
q

Ожидаемое действие
Не является квадратом

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

Вставка (0,0)(0,1)(1,1)(1,0) на
позицию 1

Печать стека
Вывод количества элементов,
площадь которых < 2 (1)

Выход

test_03.txt

2
1
0 0 1 1 2 0 1 -1
5
4
1
5
q

Ожидаемое действие
Вставка (0,0)(1,1)(2,0)(-1,1) на
позицию 1

Печать стека
Удаление элемента с
позиции 1
Печать стека
Выход

7. 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: количеств элементов с площадью меньше чем 2 :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 стек
- 2) вставить элемент на позицию
- 3) (pop)удаление верхнего элемента
- 4) (top) значение вернего элемента
- 5) удалить элемент с позиции
- 6) напечатать стек
- 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\$

8. 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.