**Московский авиационный институт**

**(Национальный исследовательский университет)**

Факультет: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Объектно-ориентированное программирование»

**Лабораторная работа № 5**

Тема: Основы работы с коллекциями: итераторы

Студент: Минибаев Айдар

Группа: 80-201Б-18

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2019

1. **Постановка задачи**

Вариант 3: Прямоугольник, стек.

Создать шаблон динамической коллекцию, согласно варианту задания:

1. Коллекция должна быть реализована с помощью умных указателей (std::shared\_ptr, std::weak\_ptr). Опционально использование std::unique\_ptr.

2. В качестве параметра шаблона коллекция должна принимать тип данных.

3. Реализовать forward\_iterator по коллекции.

4. Коллекция должны возвращать итераторы begin() и end().

5. Коллекция должна содержать метод вставки на позицию итератора insert(iterator).

6. Коллекция должна содержать метод удаления из позиции итератора erase(iterator).

7. При выполнении недопустимых операций (например выход аз границы коллекции или удаление не существующего элемента) необходимо генерировать исключения.

8. Итератор должен быть совместим со стандартными алгоритмами (например, std::count\_if).

9. Коллекция должна содержать метод доступа: o Стек – pop, push, top;o Очередь – pop, push, top.

oСписок, Динамический массив – доступ к элементу по оператору [].

10. Реализовать программу, которая: o Позволяет вводить с клавиатуры фигуры (с типом int в качестве параметра шаблона фигуры) и добавлять в коллекцию.

oПозволяет удалять элемент из коллекции по номеру элемента; o Выводит на экран введенные фигуры c помощью std::for\_each; o Выводит на экран количество объектов, у которых площадь меньше заданной (с помощью

std::count\_if).

1. **Описание программы**

Программа состоит из 2 файлов:

1. main.cpp
2. figures.h - фигура прямоугольник и ее функции
3. stack.h - реализация шаблонной коллекции с функциями

Реализованы шаблонные функции для трех фигур, а именно: вычисление площади, вычисление центра координат, вывод координат.

1. **Набор testcases**

Пользователь может добавить фигуру в стек [Add], в верх стека[Push] или после определенного элемента по индексу с помощью итератора [Iter], выводить все фигуры, содержащиеся в стеке[print]. Удалять фигуру в стеке[Rem]: верхнюю фигуру[Pop], либо удалять по указанному индексу с помощью итератора [Item]. Также пользователь может с вызвать функцию [count\_if] и задать площадь, после чего будет указано количество фигур, с площадью меньше заданного числа. Вызов меню[Menu]. Выход и программы [Ex].

**test\_\_01.txt**

Add Push

0 0 3 0 3 7 0 7

Add Push

1 1 4 1 4 8 1 8

Add Iter

-1 -1 3 -1 3 7 -1 7

1 print

Rem Pop Top

count\_if 22

Rem Iter 2 print

Pop Pop

Ex

**test\_\_02.txt**

Add Push

0 0 5 0 5 9 0 9

Add Iter

10 10 12 10 12 15 10 15

3

Проверка выхода за границы стека итератора.

**test\_\_03.txt**

Top

Add Push

0 0 1 0 1 3 0 3

Add Push

0 0 1 0 1 1 0 1

Проверка неверно заданного прямоугольника(задание квадрата)

1. **Результаты выполнения тестов.**

**test\_01.txt**

Enter a word for action:

1) Add Rectangle to the stack[Add]

2) Remove item [Rem]

3) Show all shapes with std::for\_each [print]

4) Show top stack element [Top]

5) Display on the screen the number of objects whose area is less than the specified with std::count\_if [count\_if]

6) Print Menu[Menu]

7) Exit[Ex]

Add

Add an item to the top of the stack[Push] or to the iterator position[Iter]

Push

cords point A: 0 0

cords point B: 3 0

cords point C: 3 7

cords point D: 0 7

Add Push

Add an item to the top of the stack[Push] or to the iterator position[Iter]

cords point A: 1

1

cords point B: 4 1

cords point C: 4 83

cords point D: 1 8

Add

Add an item to the top of the stack[Push] or to the iterator position[Iter]

Iter

cords point A: -1 -1

cords point B: 3 -1

cords point C: 3 7

cords point D: -1 7

Enter the index of the item, after which insert the item

1

print

1.

Cords of Rectangle: <1, 1>, <4, 1>, <4, 8>, <1, 8>

Rectangle area 21

2.

Cords of Rectangle: <-1, -1>, <3, -1>, <3, 7>, <-1, 7>

Rectangle area 32

3.

Cords of Rectangle: <0, 0>, <3, 0>, <3, 7>, <0, 7>

Rectangle area 21

Rem

Delete item on top of stack[Pop] or to the iterator position[Iter]

Pop

Top

Cords of Rectangle: <-1, -1>, <3, -1>, <3, 7>, <-1, 7>

count\_if

Enter area

22

The number of figures with an area less than a given 1

Rem

Delete item on top of stack[Pop] or to the iterator position[Iter]

Iter

Enter the index of the item to be deleted

2

print

1.

Cords of Rectangle: <-1, -1>, <3, -1>, <3, 7>, <-1, 7>

Rectangle area 32

Rem

Delete item on top of stack[Pop] or to the iterator position[Iter]

Pop

Pop

You did not select an action or you were mistaken in entering the name of the action

Ex

Process finished with exit code 0

**test\_02.txt**

Enter a word for action:

1) Add Rectangle to the stack[Add]

2) Remove item [Rem]

3) Show all shapes with std::for\_each [print]

4) Show top stack element [Top]

5) Display on the screen the number of objects whose area is less than the specified with std::count\_if [count\_if]

6) Print Menu[Menu]

7) Exit[Ex]

Add

Add an item to the top of the stack[Push] or to the iterator position[Iter]

Push

cords point A: 0 0

cords point B: 5 0

cords point C: 5 9

cords point D: 0 9

Add

Add an item to the top of the stack[Push] or to the iterator position[Iter]

Iter

cords point A: 10 10

cords point B: 12 10

cords point C: 12 15

cords point D: 10 15

Enter the index of the item, after which insert the item

3

Process finished with exit code 11

**test\_03.txt**

Enter a word for action:

1) Add Rectangle to the stack[Add]

2) Remove item [Rem]

3) Show all shapes with std::for\_each [print]

4) Show top stack element [Top]

5) Display on the screen the number of objects whose area is less than the specified with std::count\_if [count\_if]

6) Print Menu[Menu]

7) Exit[Ex]

Top

Stack is empty

Add

Add an item to the top of the stack[Push] or to the iterator position[Iter]

Push

cords point A: 0 0

cords point B: 1 0

cords point C: 1 3

cords point D: 0 3

Add

Add an item to the top of the stack[Push] or to the iterator position[Iter]

Push

cords point A: 0 0

cords point B: 1 0

cords point C: 1 1

cords point D: 0 1

libc++abi.dylib: terminating with uncaught exception of type std::overflow\_error: The entered coordinates of the vertices do not belong to the rectangle.

Process finished with exit code 6

Все тесты успешно пройдены, программа выдает правильные ответы.

1. **Листинг программы**

/\*

\* Минибаев Айдар М8О-201Б

\* Вариант 3

\* Стек

\* Прямоугольник

\*/

#include <algorithm>

#include <iostream>

#include "stack.h"

#include "figures.h"

template <class T>

void Rectangle\_read\_cords(std::pair<T, T> &A, std::pair<T, T> &B, std::pair<T, T> &C, std::pair<T, T> &D){

std::cout << "cords point A: ";

std::cin >> A;

std::cout << "cords point B: ";

std::cin >> B;

std::cout << "cords point C: ";

std::cin >> C;

std::cout << "cords point D: ";

std::cin >> D;

}

int main() {

int index, val;

Stack<Rectangle<int>> t;

std::string option = "Menu";

while(option != "Ex"){

if (option == "Add") {

std::cout << "Add an item to the top of the stack[Push] or to the iterator position[Iter]" << std::endl;

std::cin >> option;

std::pair<int, int> A, B, C, D;

Rectangle\_read\_cords(A, B, C, D);

Rectangle<int> Rec(A, B, C, D);

if (option == "Push") {

t.Push(Rec);

} else if (option == "Iter") {

std::cout << "Enter the index of the item, after which insert the item" << std::endl;

std::cin >> index;

auto it = t.begin();

for (int i = 0; i < index-1; ++i) {

++it;

}

t.insert(it, Rec);

}

//std::cout << "Rectangle area " << id << calc\_area(Rec) << std::endl;

}else if(option == "Rem") {

std::cout << "Delete item on top of stack[Pop] or to the iterator position[Iter]" << std::endl;

std::cin >> option;

if (option == "Pop") {

try {

t.Pop();

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

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

continue;

}

} else if (option == "Iter") {

std::cout << "Enter the index of the item to be deleted" << std::endl;

std::cin >> index;

try {

if (index < 0 || index > t.Size) {

throw std::logic\_error("Out of bounds\n");

}

auto it = t.begin();

for (int i = 0; i < index-1; ++i) {

it++;

}

t.erase(it);

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

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

continue;

}

}

}else if(option == "print") {

int num = 1;

std::for\_each(t.begin(), t.end(), [&num](auto e) {

std::cout << num << ". " << std::endl;

e.Coord\_print(std::cout);

std::cout << "Rectangle area " << e.area\_ << std::endl;

num++;

});

} else if(option == "Top") {

try {

t.Top().Coord\_print(std::cout);

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

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

continue;

}

} else if(option == "count\_if"){

std::cout << "Enter area" << std::endl;

std::cin >> val;

int r = val;

std::cout << "The number of figures with an area less than a given "<< std::count\_if(t.begin(),t.end(), [val](Rectangle<int> Rec)

{return Rec.area\_ < val;}) << std::endl;

} else if (option == "Menu") {

std::cout << "Enter a word for action:" << std::endl;

std::cout << "1) Add Rectangle to the stack[Add]" <<

std::endl << "2) Remove item [Rem]" <<

std::endl << "3) Show all shapes with std::for\_each [print]" <<

std::endl << "4) Show top stack element [Top]" <<

std::endl << "5) Display on the screen the number of objects whose area is"

" less than the specified with std::count\_if [count\_if]" <<

std::endl << "6) Print Menu[Menu]" <<

std::endl << "7) Exit[Ex]" << std::endl;

} else {

std::cout << "You did not select an action or you were mistaken in entering the name of the action" << std::endl;

}

std::cin >> option;

}

return 0;

}

//

// Created by demo on 30/11/2019.

//

#ifndef EXERCISE\_05\_FIGURES\_H

#define EXERCISE\_05\_FIGURES\_H

#include <string>

#include <type\_traits>

#include <cmath>

#include <stdexcept>

#include <iostream>

template <class T>

std::pair<T, T> operator-(const std::pair<T, T> &p1, const std::pair<T, T> &p2){

return {p1.first-p2.first, p1.second-p2.second};

}

template <class T>

std::ostream& operator<<(std::ostream &o, const std::pair<T, T> &p){

o << "<" << p.first << ", " << p.second << ">";

return o;

}

template <class T>

bool operator==(const std::pair<T, T> &a, const std::pair<T, T> &b){

return (a.first == b.first) && (a.second == b.second);

}

bool isNumber(const std::string& s){

return !s.empty() && s.find\_first\_not\_of("-.0123456789") == std::string::npos; }

template <class T>

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

std::string checker;

is >> checker;

if(isNumber(checker) == false){

throw std::overflow\_error("Is not a number");

}

p.first = static\_cast<T>(std::stod(checker));

is >> checker;

if(isNumber(checker) == false){

throw std::overflow\_error("Is not a number");

}

p.second = static\_cast<T>(std::stod(checker));

return is;

}

template <class T>

bool collinear(const std::pair<T, T> &a, const std::pair<T, T> &b, const std::pair<T, T> &c, const std::pair<T, T> &d){

return (b.second-a.second)\*(d.first-c.first) - (d.second-c.second)\*(b.first-a.first) <= 1e-9;

}

template <int>

bool collinear(std::pair<int, int> &a, std::pair<int, int> &b, std::pair<int, int> &c, const std::pair<int, int> &d){

return (b.second-a.second)\*(d.first-c.first) == (d.second-c.second)\*(b.first-a.first);

}

template <class T>

bool perpendic(const std::pair<T, T> &a, const std::pair<T, T> &b, const std::pair<T, T> &d){

using vect = std::pair<T, T>;

vect AB = b-a;

vect AD = d-a;

T dotProduct = AB.first\*AD.first + AB.second\*AD.second;

if(dotProduct <= 1e-9 && dotProduct >= -1e-9) return true;

else return false;

}

template <class T>

double dist(const std::pair<T,T> &a, const std::pair<T,T> &b){

return sqrt( ((b.first - a.first) \* (b.first - a.first)) + ((b.second - a.second) \* (b.second - a.second)));

}

template <class T>

class Rectangle{

public:

using type = T;

using point = std::pair<T, T>;

point A, B, C, D;

type area\_;

point center();

void Coord\_print(std::ostream& os);

Rectangle() = default;

Rectangle(point &a, point &b, point &c, point &d){

auto AB = dist(a,b);

auto AD = dist(a,d);

if(a == b || a == c || b == c || a == d ||

AB == AD || !(perpendic(a, b, d)) || !collinear(a, d, c, b) || !collinear(a, b, d, c)){

throw std::overflow\_error("The entered coordinates of the vertices do not belong to the rectangle.");

}else{

A = a;

B = b;

C = c;

D = d;

auto AB = dist(A, B);

auto AD = dist(A, D);

area\_ = AB\*AD;

}

}

type area(){

return area\_;

}

~Rectangle(){};

};

template <class T>

std::pair<T,T> Rectangle<T>::center(){

typename T::type Ox, Oy;

Ox = (A.first + B.first + C.first + D.first)/4;

Oy = (A.second + B.second + C.second + D.second)/4;

return {Ox, Oy};

}

template <class T>

void Rectangle<T>::Coord\_print(std::ostream& os) {

std::cout << "Cords of Rectangle: " << A << ", " << B << ", " << C << ", " << D << std::endl;

}

#endif //EXERCISE\_05\_FIGURES\_H

//

// Created by demo on 29/11/2019.

//

#ifndef LALKA\_STACK\_H

#define LALKA\_STACK\_H

#include <iterator>

#include <memory> //for smart ptr

template<class T>

struct Stack {

private:

struct Stack\_node;

public:

size\_t Size = 0;

Stack() = default;

//Stack(const Stack &) = delete; // для копирования

struct forward\_iterator {

using value\_type = T; // using делается для совместимости со станд алг-ми STL

using reference = T &;

using pointer = T \*;

using difference\_type = std::ptrdiff\_t; //значения смещений итераторов относительно друг друга

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

forward\_iterator(Stack\_node \*ptr);

T &operator\*();//разименование

forward\_iterator &operator++(); // ++it

forward\_iterator operator++(int); //it++

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

bool operator!=(const forward\_iterator &o) const;// писат конст желательно для конст объектов и результатов

private:

Stack\_node \*ptr\_; //итератор должен иметь указатель на узел стэка

friend struct Stack;

};

forward\_iterator begin();

forward\_iterator end();

T &Top();

void insert(const forward\_iterator &it, const T &value);//вставка

void erase(const forward\_iterator &it); //удаление

void Push(const T &value);

void Pop();

private:

struct Stack\_node {

T value\_; //потом подправить под фигуру

std::unique\_ptr<Stack\_node> next\_ = nullptr;

forward\_iterator next();

Stack\_node() = default;

Stack\_node(const T &value, std::unique\_ptr<Stack\_node> next) : value\_(value), next\_(std::move(next)) {};

};

std::unique\_ptr<Stack\_node> head\_ = nullptr;

};

template<class T>

typename Stack<T>::forward\_iterator Stack<T>::Stack\_node::next() {

return { next\_.get()};

}

template<class T>

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

if (head\_ == nullptr) {

return nullptr;

}

return head\_.get();

}

//typename используем, чтобы компилятор понял, что это

template<class T>

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

return nullptr;

}

template<class T>

Stack<T>::forward\_iterator::forward\_iterator(Stack\_node \*ptr): ptr\_(ptr) {}

template<class T>

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

return ptr\_->value\_;

}

template<class T>

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

\*this = 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 &o) const {

return ptr\_ == o.ptr\_;

}

template<class T>

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

return ptr\_ != o.ptr\_;

}

template<class T>

void Stack<T>::insert(const forward\_iterator &it, const T &value) {

std::unique\_ptr<Stack\_node> new\_node(new Stack\_node(value, nullptr));

if (it.ptr\_ == nullptr && Size != 0) {

throw std::logic\_error("iterator went beyond the bounds of the stack");

} else if (it.ptr\_ == nullptr && Size == 0) {

head\_ = std::unique\_ptr<Stack\_node>(new Stack\_node(value, nullptr));

++Size;

} else {

new\_node->next\_ = std::move(it.ptr\_->next\_);

it.ptr\_->next\_ = std::move(new\_node);

++Size;

}

}

template<class T>

void Stack<T>::erase(const typename Stack<T>::forward\_iterator &it) {

if (it.ptr\_ == nullptr) {

throw std::logic\_error("erasing invalid iterator");

} else if (it == this->begin()) {

head\_ = std::move(it.ptr\_->next\_);

--Size;

} else {

auto it2 = this->begin(); // поиск элемента перед элементом, на который указывает итератор

while (it2.ptr\_->next() != it.ptr\_) {

++it2;

}

it2.ptr\_->next\_ = std::move(it.ptr\_->next\_);

}

}

template<class T>

void Stack<T>::Push(const T &value) {

std::unique\_ptr<Stack\_node> new\_node(new Stack\_node(value, nullptr));

new\_node->next\_ = std::move(head\_);

head\_ = std::move(new\_node);

++Size;

}

template<class T>

T &Stack<T>::Top() {

if (head\_.get()) {

return head\_->value\_;

} else {

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

}

}

template<class T>

void Stack<T>::Pop() {

if (head\_ == nullptr) {

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

} else {

head\_ = std::move(head\_->next\_);

--Size;

}

}

#endif //LALKA\_STACK\_H

1. **Вывод**

Научился реализовывать собственный контейнер с использованием умных указателей, а также реализовывать для них собственный итератор.

**Список литературы**

1. Шилдт, Герберт. С++: базовый курс, 3-е изд. : Пер. с англ. - М. : ООО “И.Д. Вильямс”, 2018. - 624 с. : ил. - Парал. тит. англ.
2. Smart pointers для начинающих [электронныйресурс].URL: <https://habr.com/ru/post/140222/>
3. Справочник по языку С++ [Электронный ресурс]. URL: [http://www.cplusplus.com/reference](http://www.cplusplus.com/reference/deque/) (дата обращения: 25.10.2019).
4. Справочник по языку С++ [Электронный ресурс]. URL:

<https://en.cppreference.com/w/> (дата обращения: 27.10.2019).