**Московский авиационный институт**

**(Национальный исследовательский университет)**

Институт: «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: Хренникова Ангелина Сергеевна

Группа: 80-208

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2020

**Содержание**

1. Постановка задачи
2. Описание программы
3. Набор тестов
4. Результаты выполнения тестов
5. Листинг программы
6. Вывод
7. Список литературы

**Постановка задачи**

Цель: Изучить основы работы с коллекциями, ознакомиться с шаблоном проектирования «Итератор».

Задача: Разработать шаблоны классов фигура: пятиугольник, контейнер: стек. Параметром шаблона должен являться скалярный тип данных задающий тип данных для оси координат. Классы должны иметь публичные поля. Фигуры являются фигурами вращения, т.е. равносторонними. Для хранения координат фигур необходимо использовать шаблон std::pair.

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 - фигуры;
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 за границы коллекции или удаление несуществующего элемента) необходимо генерировать исключения;
8. Итератор должен быть совместим со стандартными алгоритмами (например, std::count\_if)
9. Коллекция должна содержать метод доступа: стек – pop, push, top;
10. Реализовать программу, которая:

* Позволяет вводить с клавиатуры фигуры (с типом int в качестве параметра шаблона фигуры) и добавлять в коллекцию;
* Позволяет удалять элемент из коллекции по номеру элемента;
* Выводит на экран введенные фигуры c помощью std::for\_each;
* Выводит на экран количество объектов, у которых площадь меньше заданной (с помощью std::count\_if).

**Описание программы**

Пользователю доступны действия: push (добавить элемент), pop (удалить элемент), print (распечатать стек), insert (вставить пятиугольник на конкретную позицию), erase (удалить пятиугольник с конкретной позиции), count (подсчитать количество пятиугольников, чья площадь меньше заданной). В программе есть специальное меню, позволяющее выбрать определенное действие. В случае некорректно введенных данных генерируются исключения.

Пятиугольник задается координатими центральной точкой и одной из точек самой фигуры, остальные точки вычисляются программой. Для каждой фигуры печатаются координаты и площадь.

В программе реализован класс пятиугольника, для работы с фигурой(полями являются пары координат и площадь), класс стека, класс итераторов. В реализации стека учтены все основные функции коллекции, такие как pop, push, top и insert, erase. Позиции в стеке начинаются с 0.

**Набор тестов**

Тест 1: 1 0 0 0 6 1 0 0 0 5 4 1 0 0 0 3 4 0 0 0 0 1 3 2 6 25 0

1. В стек добавляется пятиугольник с центром в точке (0;0) и точкой (0;6);
2. В стек добавляется пятиугольник с центром в точке (0;0) и точкой (0;5);
3. В стек добавляется пятиугольник с центром в точке (0;0) и точкой (0;3) на позицию 1;
4. В стек добавляется пятиугольник с центром в точке (0;0) и точкой (0;1) на позицию 0;
5. Печатается весь стек;
6. Удаляется элемент из стека;
7. Считается количество элементов в стеке, площадь которых меньше 25;
8. Программа завершается.

Тест 2: 1 0 0 1 1 1 0 0 2 2 4 0 0 0 3 3 1 0 0 4 4 5 2 3 6 50 0

1. В стек добавляется пятиугольник с центром в точке (0;0) и точкой (1;1);
2. В стек добавляется пятиугольник с центром в точке (0;0) и точкой (2;2);
3. В стек добавляется пятиугольник с центром в точке (0;0) и точкой (3;3) на позицию 0;
4. В стек добавляется пятиугольник с центром в точке (0;0) и точкой (4;4);
5. Удаляется элемент на 2 позиции стека;
6. Печатается весь стек;
7. Считается количество элементов в стеке, площадь которых меньше 50;
8. Программа завершается.

**Результат выполнения тестов**

Тест 1:

Menu:

1. Push figure to stack

2. Pop figure from stack

3. Print stack

4. Insert item to pos N

5. Delete item from pos N

6. Count figures with area less then area

0. Exit

Select menu item: 1

Enter figure cords: 0 0 0 6

Select menu item: 1

Enter figure cords: 0 0 0 5

Select menu item: 4

Enter position: 1

Enter figure cords: 0 0 0 3

Select menu item: 4

Enter position: 0

Enter figure cords: 0 0 0 1

Select menu item: 3

[(0, 1), (-0.951057, 0.309017), (-0.587785, -0.809017), (0.587785, -0.809017), (0.951057, 0.309017)]

Area: 2.37764

[(0, 5), (-4.75528, 1.54508), (-2.93893, -4.04508), (2.93893, -4.04508), (4.75528, 1.54508)]

Area: 59.441

[(0, 3), (-2.85317, 0.927051), (-1.76336, -2.42705), (1.76336, -2.42705), (2.85317, 0.927051)]

Area: 21.3988

[(0, 6), (-5.70634, 1.8541), (-3.52671, -4.8541), (3.52671, -4.8541), (5.70634, 1.8541)]

Area: 85.5951

Select menu item: 2

Select menu item: 6

Enter area: 25

Result = 1

Select menu item: 0

Тест 2:

Menu:

1. Push figure to stack

2. Pop figure from stack

3. Print stack

4. Insert item to pos N

5. Delete item from pos N

6. Count figures with area less then area

0. Exit

Select menu item: 1

Enter figure cords: 0 0 1 1

Select menu item: 1

Enter figure cords: 0 0 2 2

Select menu item: 4

Enter position: 0

Enter figure cords: 0 0 3 3

Select menu item: 1

Enter figure cords: 0 0 4 4

Select menu item: 5

Enter position: 2

Select menu item: 3

[(4, 4), (-2.56816, 5.04029), (-5.58721, -0.884927), (-0.884927, -5.58721), (5.04029, -2.56816)]

Area: 76.0845

[(3, 3), (-1.92612, 3.78022), (-4.19041, -0.663695), (-0.663695, -4.19041), (3.78022, -1.92612)]

Area: 42.7975

[(1, 1), (-0.64204, 1.26007), (-1.3968, -0.221232), (-0.221232, -1.3968), (1.26007, -0.64204)]

Area: 4.75528

Select menu item: 6

Enter area: 50

Result = 2

Select menu item: 0

**Листинг программы**

/\*Вариант - 6

Хренникова Ангелина

Фигура: пятиугольник; Контейнер: стек;

\*/

#include <iostream>

#include <string>

#include <cmath>

#include <algorithm>

#include <memory>

#include <exception>

const double PI = 3.14159265358;

template <class T>

struct node { //Класс элемента

using type = T;

node() {};

node(T key) : value(key) {}; //Элемент

std::shared\_ptr<node<T>> next; //Указатель на следующий элемент

std::weak\_ptr<node<T>> prev; //Указатель на предыдущий элемент

T value;

};

template <class T>

class iterator { //Класс итераторов

protected:

iterator() {};

public:

typedef std::bidirectional\_iterator\_tag iterator\_category;

typedef T value\_type;

typedef int difference\_type;

typedef T\* pointer;

typedef T& reference;

iterator(std::shared\_ptr<T> pointer) : ptr(pointer) {}; //Конструктор

iterator operator++() { //Следующий итератор

ptr = (ptr.lock())->next;

return \*this;

}

iterator operator++(int junk) { //Следующий итератор

iterator i = \*this;

ptr = (ptr.lock())->next;

return i;

}

iterator operator--() { //Предыдущий итератор

ptr = (ptr.lock())->prev;

return \*this;

}

iterator operator--(int junk) { //Предыдущий итератор

iterator i = \*this;

ptr = (ptr.lock())->prev;

return i;

}

typename T::type& operator\*() { //Значение

return (ptr.lock())->value;

}

std::shared\_ptr<T> operator->() { //Указатель

return ptr.lock();

}

bool operator==(const iterator& rhs) { //Равенство

return ptr.lock() == rhs.ptr.lock();

}

bool operator!=(const iterator& rhs) { //Отсутствие равенства

return ptr.lock() != rhs.ptr.lock();

}

private:

std::weak\_ptr<T> ptr; //Указалель на элемент

};

template <class T>

class stack : public iterator<node<T>> { //Класс стека

public:

stack() { //Конструктор

fake = std::make\_shared<node<T>>();

fake->next = fake;

fake->prev = fake;

size\_ = 0;

}

~stack() { //Деструктор

while (fake->next != fake) {

pop();

}

fake->next = nullptr;

}

void push(T item) { //Добавление в стек

auto temp = std::make\_shared<node<T>>(item);

if (fake->next == fake) {

fake->next = temp;

fake->prev = temp;

temp->next = fake;

temp->prev = fake;

}

else {

auto first = fake->next;

temp->next = first;

temp->prev = fake;

first->prev = temp;

fake->next = temp;

}

size\_++;

}

void pop() { //Удаление из стека

if (is\_empty()) {

throw std::logic\_error("Empty stack pop");

}

if (fake->next == (fake->prev).lock()) {

fake->next = fake;

}

else {

fake->next = fake->next->next;

fake->next->prev = fake;

}

size\_--;

}

T top() const { //Верхний элемент

if (is\_empty()) {

throw std::logic\_error();

}

return fake->next->value;

}

void insert(iterator<node<T>> it, T key) { //Вставка на позицию

if (is\_empty()) {

push(key);

}

else {

auto temp = std::make\_shared<node<T>>(key);

temp->next = (it->next->prev).lock();

temp->prev = it->prev;

((it->prev).lock())->next = temp;

it->prev = temp;

size\_++;

}

}

void erase(iterator<node<T>> it) { //Удаление с позиции

if (is\_empty()) {

throw std::logic\_error("Empty stack erase");

}

if (it == end()) {

throw std::logic\_error("Removing fake item is permitted");

}

it->next->prev = it->prev;

((it->prev).lock())->next = it->next;

}

bool is\_empty() const { //Проверка на пустоту

return size\_ == 0;

}

size\_t size() const { //Возвращение размера

return size\_;

}

iterator<node<T>> begin() { //Начало стека

return iterator<node<T>>(fake->next);

}

iterator<node<T>> end() { //Конец стека

return iterator<node<T>>(fake);

}

iterator<node<T>> at(int pos) { //Не попадание в стек

if (pos >= size\_ || pos < 0) {

throw std::logic\_error("Wrong position iterator");

}

if (is\_empty()) {

throw std::logic\_error("Can not get iterator from empty stack");

}

int k = 0;

if (pos < size\_ / 2) {

iterator<node<T>> temp(begin());

while (k < pos) {

temp++;

k++;

}

return temp;

}

else {

iterator<node<T>> temp(--end());

while (size\_ - k - 1 > pos) {

temp--;

k++;

}

return temp;

}

}

void print() { //Печать стека

std::cout << "stack: ";

for (auto i = begin(); i != end(); i++) {

std::cout << \*i << '\n';

}

std::cout << "\n";

}

public:

std::shared\_ptr<node<T>> fake; //Указатель на элемент

size\_t size\_; //Размер стека

};

template <class T>

std::istream& operator>>(std::istream& is, std::pair<T, T>& p) { //Вывод пары координат

is >> p.first >> p.second;

return is;

}

template <class T>

class pentagon { //Класс пятиугольника

public:

friend std::ostream& operator<<(std::ostream& os, pentagon<T>& p) { //Вывод пятиугольника

os << "[";

for (int i = 1; i < 5; i++) {

os << "(" << p.cords[i].first << ", " << p.cords[i].second << "), ";

}

os << "(" << p.cords[5].first << ", " << p.cords[5].second << ")";

os << "]\n";

os << "Area: " << p.area\_ ;

return os;

}

friend std::istream& operator>>(std::istream& is, pentagon<T>& p) { //Ввод пятиугольника

for (int i = 0; i < 2; i++) {

is >> p.cords[i];

}

double c = (sqrt(5) - 1) / 4;

double s = (sqrt(10 + 2 \* sqrt(5))) / 4;

p.cords[2].first = p.cords[0].first + (p.cords[1].first - p.cords[0].first) \* c - (p.cords[1].second - p.cords[0].second) \* s;

p.cords[2].second = p.cords[0].second + (p.cords[1].first - p.cords[0].first) \* s + (p.cords[1].second - p.cords[0].second) \* c;

p.cords[3].first = p.cords[0].first + (p.cords[2].first - p.cords[0].first) \* c - (p.cords[2].second - p.cords[0].second) \* s;

p.cords[3].second = p.cords[0].second + (p.cords[2].first - p.cords[0].first) \* s + (p.cords[2].second - p.cords[0].second) \* c;

p.cords[4].first = p.cords[0].first + (p.cords[3].first - p.cords[0].first) \* c - (p.cords[3].second - p.cords[0].second) \* s;

p.cords[4].second = p.cords[0].second + (p.cords[3].first - p.cords[0].first) \* s + (p.cords[3].second - p.cords[0].second) \* c;

p.cords[5].first = p.cords[0].first + (p.cords[4].first - p.cords[0].first) \* c - (p.cords[4].second - p.cords[0].second) \* s;

p.cords[5].second = p.cords[0].second + (p.cords[4].first - p.cords[0].first) \* s + (p.cords[4].second - p.cords[0].second) \* c;

double r = sqrt(pow(p.cords[1].first - p.cords[0].first, 2) + pow(p.cords[1].second - p.cords[0].second, 2));

p.area\_ = 0.5 \* r \* r \* 5 \* s;

return is;

}

pentagon() {}; //Контструктор

pentagon(std::pair<T, T> p0, std::pair<T, T> p1, std::pair<T, T> p2, std::pair<T, T> p3,

std::pair<T, T> p4, std::pair<T, T> p5) { //Конструктор

cords[0] = p0;

cords[1] = p1;

cords[2] = p2;

cords[3] = p3;

cords[4] = p4;

cords[5] = p5;

area\_ = 0;

double s = (sqrt(10 + 2 \* sqrt(5))) / 4;

double r = sqrt(pow(cords[1].first - cords[0].first, 2) + pow(cords[1].second - cords[0].second, 2));

area\_ = 0.5 \* r \* r \* 5 \* s;

}

double area() { //Возвращение площади

return area\_;

}

~pentagon() {}; //Деструктор

private:

std::pair<T, T> cords[6]; //Координаты

double area\_; //Площадь

};

void menu() {

std::cout << "Menu: \n";

std::cout << "1. Push figure to stack\n"; //Добавление в стек

std::cout << "2. Pop figure from stack\n"; //Удаление из стека

std::cout << "3. Print stack\n"; //Печать стека

std::cout << "4. Insert item to pos N\n"; //Вставка на позицию

std::cout << "5. Delete item from pos N\n"; //Удаление с позиции

std::cout << "6. Count figures with area less then area\n"; //Подсчет, сколько фигур имеют площадь, меньше заданной

std::cout << "0. Exit\n"; //Выход из программы

}

int main() {

stack<pentagon<double>> st;

menu();

int action = 0;

std::cout << "Select menu item: ";

std::cin >> action;

while (action > 0 && action < 7) {

if (action == 1) { //Вставка в стек

std::cout << "Enter figure cords: ";

pentagon<double> temp;

std::cin >> temp;

st.push(temp);

}

if (action == 2) { //Удаление из стека

try {

st.pop();

}

catch (std::exception& ex) {

std::cout << "Exception with pop(): " << ex.what() << "\n";

}

}

if (action == 3) { //Печать стека

std::for\_each(st.begin(), st.end(), [](pentagon<double>& p) { std::cout << p << "\n";});

}

if (action == 4) { //Вставка на позицию

int k;

std::cout << "Enter position: ";

std::cin >> k;

try {

auto it = st.at(k);

std::cout << "Enter figure cords: ";

pentagon<double> temp;

std::cin >> temp;

st.insert(it, temp);

}

catch (std::exception& ex) {

std::cout << "Exception with insert(): " << ex.what() << "\n";

}

}

if (action == 5) { //Удаление с позиции

int k;

std::cout << "Enter position: ";

std::cin >> k;

try {

auto it = st.at(k);

st.erase(it);

}

catch (std::exception& ex) {

std::cout << "Exception with erase(): " << ex.what() << "\n";

}

}

if (action == 6) { //Подсчет, сколько фигур имеют площадь, меньше заданной

double k;

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

std::cin >> k;

int ans = std::count\_if(st.begin(), st.end(), [k](pentagon<double>& pd) -> bool{ return pd.area() < k; });

std::cout << "Result = " << ans << "\n";

}

std::cout << "Select menu item: ";

std::cin >> action;

}

return 0;

}

**Выводы**

В результате выполнения данной лабораторной работы я получила навыки реализации шаблонных контейнеров, а также научилась работать с умными указателями и создавать свои итераторы.

Умные указатели - полезная вещь, так как они при выходе из области видимости сами удаляют то, на что указывали, и поэтому они позволяют избежать утечек памяти.

Главное предназначение итераторов заключается в предоставлению возможности пользователю обращаться к любому элементу контейнера при сокрытии внутренней структуры от пользователя. Это позволяет контейнеру хранить элементы любым способом при допустимости работы пользователя с ним как с простой последовательностью или списком.

**Список литературы**

1. Стандартные библиотеки шаблонов/Электронный ресурс/Режим доступа: <https://tproger.ru/articles/stl-cpp/> (дата обращения: 20.11.2020).
2. Справочник по языку С++/Электронный ресурс/Режим доступа: <https://en.cppreference.com/w/> (дата обращения: 22.11.2020).
3. C++ reference - std::iterator\_traits/Электронный ресурс/Режим доступа: [std::iterator\_traits - cppreference.com](https://en.cppreference.com/w/cpp/iterator/iterator_traits) (дата обращения: 22.11.2020).