**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 метапрограммирования

Студент: Калугин Кирилл Алексеевич

Группа: 80-207

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2020

1. Постановка задачи

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 аз 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;
* очередь – pop, push, top;
* список, Динамический массив – доступ к элементу по оператору [].

10. Реализовать программу, которая:

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

Вариант: пятиугольник, стек**.**

1. Описание программы

Программа реализует стек при помощи умных указателей. Каждый элемент хранит ссылку лишь на следующий элемент. Также реализован нестандартный для стэка итератор, имеющий возможность перемещаться по всей коллекции и вставлять, либо удалять элементы по месту нахождения. Взаимодействие пользователя с программой осуществляется при помощи меню. Все дополнительные запросы объявляются.

1. Набор тестов

PS C:\Users\Кирилл\Desktop\ООП> .\a.exe

1.Execute command "push"

2.Execute command "pop"

3.Execute command "top"

4.Show stack

5.Move iterator to the next element

6.Move iterator to the first element

7.Move iterator to the last element

8.Add element in iterator position

9.Erase element in iterator position

10.Print the number of objects with an area less than the specified one

0.Exit

Choose option: 1

Insert center coordinates: 0 0

Insert radius: 1

Choose option: 3

----------

A = (0,1)

B = (0.882948,0.469472)

C = (0.71934,-0.694658)

D = (-0.438371,-0.898794)

E = (-0.990268,0.139173)

----------

Choose option: 2

Choose option: 2

Stack is empty!

Choose option: 1

Insert center coordinates: 1 1

Insert radius: 2

Choose option: 4

vvvvvvvvvv

----------

A = (1,3)

B = (2.7659,1.93894)

C = (2.43868,-0.389317)

D = (0.123258,-0.797588)

E = (-0.980536,1.27835)

----------

^^^^^^^^^^

Choose option: 8

Insert center coordinates: 2 2

Insert radius: 3

Choose option: 4

----------

A = (1,3)

B = (2.7659,1.93894)

C = (2.43868,-0.389317)

D = (0.123258,-0.797588)

E = (-0.980536,1.27835)

----------

vvvvvvvvvv

----------

A = (2,5)

B = (4.64884,3.40841)

C = (4.15802,-0.0839751)

D = (0.684887,-0.696382)

E = (-0.970804,2.41752)

----------

^^^^^^^^^^

Choose option: 6

Choose option: 4

vvvvvvvvvv

----------

A = (1,3)

B = (2.7659,1.93894)

C = (2.43868,-0.389317)

D = (0.123258,-0.797588)

E = (-0.980536,1.27835)

----------

^^^^^^^^^^

----------

A = (2,5)

B = (4.64884,3.40841)

C = (4.15802,-0.0839751)

D = (0.684887,-0.696382)

E = (-0.970804,2.41752)

----------

Choose option: 9

Choose option: 4

vvvvvvvvvv

----------

A = (2,5)

B = (4.64884,3.40841)

C = (4.15802,-0.0839751)

D = (0.684887,-0.696382)

E = (-0.970804,2.41752)

----------

^^^^^^^^^^

Choose option: 8

Insert center coordinates: 3 3

Insert radius: 4

Choose option: 4

----------

A = (2,5)

B = (4.64884,3.40841)

C = (4.15802,-0.0839751)

D = (0.684887,-0.696382)

E = (-0.970804,2.41752)

----------

vvvvvvvvvv

----------

A = (3,7)

B = (6.53179,4.87789)

C = (5.87736,0.221367)

D = (1.24652,-0.595176)

E = (-0.961072,3.55669)

----------

^^^^^^^^^^

Choose option: 10

Insert max square: 100

There are 2 pentagons with square less than 100

1. Листинг программы

/\*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 аз 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;

очередь – pop, push, top;

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

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).\*/

/\*Вариант: Пятиугольник, Стэк\*/

#include <iostream>

#include <algorithm>

#include <math.h>

#include <memory>

#include <windows.h>

using namespace std;

const double PI = 3.141592653589793238462643383279502884197169399375105820974944;

double Square;//глобальная переменная для хранения площади

template <typename T>

class Five {//класс пятиугольника

public:

using type = T;

pair<T, T> O;

T R;

Five () {};//конструктор класса

Five (T x, T y, T r): O(x, y), R(r) {};//конструктор класса

};

double square (Five <double> x) {//функция подсчета площади пятиугольника

return (2.5 \* pow (x.R, 2) \* sin (72.0 / 180.0 \* PI));

}

template <typename T>//шаблон функции поиска вершин пятиугольника

void points (Five <T> x) {

cout << "----------\n"<< "A = (" << (x.O).first << "," << (x.O).second + x.R << ")\n" << "B = ("

<< (x.O).first + x.R \* cos (28.0 / 180.0 \* PI) << "," << (x.O).second + x.R \* sin (28.0 / 180.0 \* PI)

<< ")\n" << "C = (" << (x.O).first + x.R \* cos (- 44.0 / 180.0 \* PI) << ","

<< (x.O).second + x.R \* sin (- 44.0 / 180.0 \* PI) << ")\n" << "D = ("

<< (x.O).first + x.R \* cos (- 116.0 / 180.0 \* PI) << "," << (x.O).second + x.R \* sin (- 116.0 / 180.0 \* PI)

<< ")\n" << "E = (" << (x.O).first + x.R \* cos (- 188.0 / 180.0 \* PI)

<< "," << (x.O).second + x.R \* sin (- 188.0 / 180.0 \* PI) << ")\n" << "----------\n";

}

template <typename T>

class Elem {//класс элемента стека

public:

Five <T> vault;

shared\_ptr <Elem <T>> prev;

Elem () {};//конструктор класса

Elem (T x, T y, T r, shared\_ptr <Elem <T>> p) {//конструктор класса

vault.O.first = x;

vault.O.second = y;

vault.R = r;

prev = p;

};

};

template <typename T>

class Iter {//класс итератора

public:

weak\_ptr <Elem <T>> iter;

Iter () {};//конструктор класса

Iter (weak\_ptr <Elem <T>> c) {//конструктор класса

iter = c;

}

Iter <T>\* operator++ () {//перегруженный оператор ++

shared\_ptr <Elem <T>> buffer = iter.lock();

if (buffer->prev) {

iter = buffer->prev;

} else {

throw "Iterator out of range";

}

return (this);

}

Five <T> operator\* () {//перегруженный оператор разименования

shared\_ptr <Elem <T>> buffer = iter.lock();

return (buffer->prev->vault);

}

bool operator!= (Iter <T> sec) {//перегруженный оператор неравенства

shared\_ptr <Elem <T>> b1 = iter.lock();

shared\_ptr <Elem <T>> b2 = sec.iter.lock();

bool f;

if (b1.get() != b2.get()) {

f = 1;

}

else {

f = 0;

}

return (f);

}

};

template <typename T>

class Stack {//класс стэка

public:

shared\_ptr <Elem <T>> first;

weak\_ptr <Elem <T>> last;

Iter <T> point;

shared\_ptr <Elem <T>> null;

void push () {//функция добавления элемента в начало стэка

T x;

T y;

T r;

cout << "Insert center coordinates: ";

cin >> x;

cin >> y;

cout << "Insert radius: ";

cin >> r;

if (!first) {

shared\_ptr <Elem <T>> curr (new Elem <T> (x, y, r, first));

shared\_ptr <Elem <T>> n (new Elem <T> ());

null = n;

null->prev = curr;

} else {

shared\_ptr <Elem <T>> curr (new Elem <T> (x, y, r, first->prev));

null->prev = curr;

}

point.iter = null->prev;

first = null;

if (!last.lock()) {

last = first->prev;

}

}

void pop () {//функция удаления элемента из начала стэка

if (!first) {

throw "Stack is empty!\n";

} else {

if (!first->prev) {

throw "Stack is empty!\n";

} else {

first->prev = first->prev->prev;

}

}

}

void top () {//функция просмотра элемента из начала стэка

if (!first) {

cout << "Stack is empty!\n";

} else {

if (!first->prev) {

cout << "Stack is empty!\n";

} else {

points (first->prev->vault);

}

}

}

Iter <T> begin () {//функция возврата в начало стэка

return (Iter <T> (first));

}

Iter <T> end () {//функция перемещения в конец стэка

Iter <T> j;

j.iter = last;

return (j);

}

void insert () {//функция вставки элемента по расположению итератора

T x;

T y;

T r;

cout << "Insert center coordinates: ";

cin >> x;

cin >> y;

cout << "Insert radius: ";

cin >> r;

if (first) {

shared\_ptr <Elem <T>> curr (new Elem <T> (x, y, r, point.iter.lock()->prev));

point.iter.lock()->prev = curr;

point.iter = curr;

if (!curr->prev) {

last = curr;

}

} else {

shared\_ptr <Elem <T>> curr (new Elem <T> (x, y, r, first));

shared\_ptr <Elem <T>> n (new Elem <T> ());

null = n;

null->prev = curr;

first = null;

if (!last.lock()) {

last = first->prev;

}

point.iter = curr;

}

}

void erase () {//функция удаления элемента по расположению итератора

if (first && first->prev) {

Iter <T> e;

e.iter = first;

while ((e.iter.lock()->prev).get() != point.iter.lock().get()) {

++ e;

}

e.iter.lock()->prev = point.iter.lock()->prev;

point.iter = e.iter.lock()->prev;

} else {

throw "Stack is empty!\n";

}

}

Iter <T> r () {//функция нахождения элемента, стоящего перед итератором

Iter <T> p;

p.iter = first;

while ((p.iter.lock()->prev).get() != (point.iter.lock()).get()) {

++ p;

}

return (p);

}

Iter <T> r2 () {//функция нахождения итератора

Iter <T> p2 = point;

return (p2);

}

};

template <typename T>

bool s (Five <T> f) {//функция сравнения площади с максимальной

bool ret;

if (square (f) < Square) {

ret = 1;

} else {

ret = 0;

}

return (ret);

}

template <class InputIt, class UnaryPredicate>

int Count\_if (InputIt first, InputIt last, UnaryPredicate p)//стандартная функция count\_if

{

int ret = 0;

for (; first != last; ++ first) {

if (p (\*first)) {

ret ++;

}

}

return ret;

}

int main() {

Stack <double> S;

int m = 99;

cout << "1.Execute command \"push\"\n";

cout << "2.Execute command \"pop\"\n";

cout << "3.Execute command \"top\"\n";

cout << "4.Show stack\n";

cout << "5.Move iterator to the next element\n";

cout << "6.Move iterator to the first element\n";

cout << "7.Move iterator to the last element\n";

cout << "8.Add element in iterator position\n";

cout << "9.Erase element in iterator position\n";

cout << "10.Print the number of objects with an area less than the specified one\n";

cout << "0.Exit\n";

while (m != 0) {

cout << "Choose option: ";

cin >> m;

switch (m) {//меню

case 1: {

try {

S.push ();

}

catch (const char\* exc) {

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hConsole, (WORD) ((0 << 4) | 4));

cerr << exc << endl;

SetConsoleTextAttribute(hConsole, (WORD) ((0 << 4) | 7));

}

break;

}

case 2: {

try {

S.pop ();

}

catch (const char\* exc) {

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hConsole, (WORD) ((0 << 4) | 4));

cerr << exc << endl;

SetConsoleTextAttribute(hConsole, (WORD) ((0 << 4) | 7));

}

break;

}

case 3: {

try {

S.top ();

}

catch (const char\* exc) {

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hConsole, (WORD) ((0 << 4) | 4));

cerr << exc << endl;

SetConsoleTextAttribute(hConsole, (WORD) ((0 << 4) | 7));

}

break;

}

case 4: {

try {

if ((S.first) && (S.first->prev)) {

for\_each (S.begin(), S.r(), points <double>);

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hConsole, (WORD) ((0 << 4) | 4));

cout << "vvvvvvvvvv\n";

SetConsoleTextAttribute(hConsole, (WORD) ((0 << 4) | 7));

points <double> (\*(S.r()));

SetConsoleTextAttribute(hConsole, (WORD) ((0 << 4) | 4));

cout << "^^^^^^^^^^\n";

SetConsoleTextAttribute(hConsole, (WORD) ((0 << 4) | 7));

for\_each (S.r2(), S.end(), points <double>);

} else {

cout << "Stack is empty!\n";

}

}

catch (const char\* exc) {

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hConsole, (WORD) ((0 << 4) | 4));

cerr << exc << endl;

SetConsoleTextAttribute(hConsole, (WORD) ((0 << 4) | 7));

}

break;

}

case 5: {

try {

++ S.point;

}

catch (const char\* exc) {

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hConsole, (WORD) ((0 << 4) | 4));

cerr << exc << endl;

SetConsoleTextAttribute(hConsole, (WORD) ((0 << 4) | 7));

}

break;

}

case 6: {

try {

S.point = S.begin();

++ S.point;

}

catch (const char\* exc) {

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hConsole, (WORD) ((0 << 4) | 4));

cerr << exc << endl;

SetConsoleTextAttribute(hConsole, (WORD) ((0 << 4) | 7));

}

break;

}

case 7: {

try {

S.point = S.end();

}

catch (const char\* exc) {

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hConsole, (WORD) ((0 << 4) | 4));

cerr << exc << endl;

SetConsoleTextAttribute(hConsole, (WORD) ((0 << 4) | 7));

}

break;

}

case 8: {

try {

S.insert ();

}

catch (const char\* exc) {

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hConsole, (WORD) ((0 << 4) | 4));

cerr << exc << endl;

SetConsoleTextAttribute(hConsole, (WORD) ((0 << 4) | 7));

}

break;

}

case 9: {

try {

S.erase ();

}

catch (const char\* exc) {

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hConsole, (WORD) ((0 << 4) | 4));

cerr << exc << endl;

SetConsoleTextAttribute(hConsole, (WORD) ((0 << 4) | 7));

}

break;

}

case 10: {

try {

cout << "Insert max square: ";

cin >> Square;

int k;

k = Count\_if (S.begin(), S.end(), s <double>);

cout << "There are ";

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hConsole, (WORD) ((0 << 4) | 2));

cout << k;

SetConsoleTextAttribute(hConsole, (WORD) ((0 << 4) | 7));

cout << " pentagons with square less than " << Square << "\n";

}

catch (const char\* exc) {

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hConsole, (WORD) ((0 << 4) | 4));

cerr << exc << endl;

SetConsoleTextAttribute(hConsole, (WORD) ((0 << 4) | 7));

}

break;

}

case 0: {

break;

}

default: {

cout << "Wrong command!\n";

break;

}

}

}

return 0;

}

1. Вывод

Литература

# std::for\_each[Электронный ресурс]

URL:https://en.cppreference.com/w/cpp/algorithm/for\_each дата обращения (25.11.2020)

# std::count, std::count\_if[Электронный ресурс]

URL:https://en.cppreference.com/w/cpp/algorithm/count дата обращения (25.11.2020)

# Smart pointers для начинающих[Электронный ресурс]

URL:https://habr.com/ru/post/140222/ дата обращения (25.11.2020)

# Википедия: Умный указатель[Электронный ресурс]

URL:https://ru.wikipedia.org/wiki/Умный\_указатель дата обращения (25.11.2020)