**Московский авиационный институт**

**(Национальный исследовательский университет)**

Институт: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Объектно-ориентированное программирование»

**Лабораторная работа № 3**

Тема: Механизмы наследования в С++

Студент: Лисин Роман Сергеевич

Группа: 80-206

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2021

1. **Постановка задачи**

## Разработать классы Square, Triangle, Rectangle (вариант 9), классы должны наследоваться от базового класса Figure.

Все классы должны поддерживать набор общих методов:

1. Вычисление геометрического центра фигуры.
2. Вывод в стандартный поток вывода std::cout координат вершин фигуры;
3. Вычисление площади фигуры.

Программа должна иметь следующие возможности:

* вводить из стандартного ввода std::cin фигуры, согласно варианту задания.
* сохранять созданные фигуры в динамический массив std::vector<Figure\*>.
* вызывать для всего массива общие функции (1-3 см. выше), т.е. распечатывать для каждой фигуры в массиве геометрический центр, координаты вершин и площадь.
* вычислять общую площадь фигур в массиве.
* удалять из массива фигуру по индексу.

1. **Описание программы**

Программа запрашивает на ввод 4 точки прямоугольника, 2 точки диагонали квадрата, 3 точки треугольника, кладёт их в вектор, проходится по вектору, вызывая функционал класса по заданию, далее удаляет первый элемент из вектора и снова проходится по нему.

1. **Набор тестов**

1)

1 1 1 3 4 3 4 1

1 1 3 3

1 1 2 3 3 1

2)

-2 0 -2 2 2 2 2 0

-6 1 -3 4

4 2 5 4 3 4

1. **Результаты выполнения тестов**

Введите 4 точки прямоугольника в порядке обхода в формате x1 y1:

1 1 1 3 4 3 4 1

Введите 2 точки диагонали квадрата в формате x1 y1:

1 1 3 3

Введите 3 точки треугольника в порядке обхода в формате x1 y1:

1 1 2 3 3 1

Фигуры в векторе:

Rectangle:

First point: (1,1)

Second point: (1,3)

Third point: (4,3)

Fourth point: (4,1)

Площадь = 6

Центр x = 2.5, y = 2

Triangle:

First point: (1,1)

Second point: (2,3)

Third point: (3,1)

Площадь = 2

Центр x = 2, y = 1.66667

Square:

First point: (1,1)

Second point: (3,3)

Площадь = 4

Центр x = 2, y = 2

Сумма всех площадей в векторе: 12

Удаляем элемент с индексом 0

Triangle:

First point: (1,1)

Second point: (2,3)

Third point: (3,1)

Площадь = 2

Центр x = 2, y = 1.66667

Square:

First point: (1,1)

Second point: (3,3)

Площадь = 4

Центр x = 2, y = 2

Сумма всех площадей в векторе после удаления первого элемента: 6

1. **Листинг программы**

#include <iostream>

#include <vector>

#include <clocale>

#include <cmath>

using namespace std;

/\*Лабораторная работа 3, подготовил Лисин Роман, группа М8О-206Б-20

Вариант 9: Треугольник, Квадрат, Прямоугольник

\*/

struct Point {

Point() {}

Point(double const& x\_, double const& y\_) : x(x\_), y(y\_) {}

double x;

double y;

};

void print\_error() {

cout << "Данные некорректны" << '\n';

}

double distance(const Point& p1, const Point& p2) {

return sqrt((p1.x - p2.x) \* (p1.x - p2.x) + (p1.y - p2.y) \* (p1.y - p2.y));

}

class Figure {

public:

virtual Point center() = 0;

virtual void print\_figure() const = 0;

virtual double area() = 0;

virtual ~Figure() {}

};

class Triangle : public Figure {

Point p1;

Point p2;

Point p3;

public:

Triangle() {}

Triangle(Point const& p1\_, Point const& p2\_, Point const& p3\_) : p1(p1\_), p2(p2\_), p3(p3\_) {}

void get\_coordinates(const Point& p1, const Point& p2, const Point& p3) {

this->p1 = p1;

this->p2 = p2;

this->p3 = p3;

}

Point center() {

return Point((this->p1.x + this->p2.x + this->p3.x) / 3, (this->p1.y + this->p2.y + this->p3.y) / 3);

}

double area() {

double square = abs((this->p2.x - this->p1.x) \* (this->p3.y - this->p1.y) - (this->p3.x - this->p1.x) \* (this->p2.y - this->p1.y)) / 2;

return square;

}

void print\_figure() const {

cout << '\n';

cout << "Triangle:" << '\n';

cout << "First point: (" << this->p1.x << "," << this->p1.y << ")" << '\n';

cout << "Second point: (" << this->p2.x << "," << this->p2.y << ")" << '\n';

cout << "Third point: (" << this->p3.x << "," << this->p3.y << ")" << '\n';

}

};

class Square : public Figure { // Квадрат задается 2 точками одной из диагоналей

Point p1;

Point p2;

public:

Square() {}

Square(Point const& p1\_, Point const& p2\_) : p1(p1\_), p2(p2\_) {}

void get\_point1(const Point& p1) {

this->p1 = p1;

}

void get\_point2(const Point& p2) {

this->p2 = p2;

}

Point center() {

return Point((p1.x + p2.x) / 2, (p1.y + p2.y) / 2);

}

double area() {

return (p1.x - p2.x) \* (p1.x - p2.x);

}

void print\_figure() const {

cout << '\n';

cout << "Square:" << '\n';

cout << "First point: (" << this->p1.x << "," << this->p1.y << ")" << '\n';

cout << "Second point: (" << this->p2.x << "," << this->p2.y << ")" << '\n';

}

};

class Rectangle : public Figure {

Point p1;

Point p2;

Point p3;

Point p4;

public:

Rectangle() {}

Rectangle(Point const& p1\_, Point const& p2\_, Point const& p3\_, Point const& p4\_) : p1(p1\_), p2(p2\_), p3(p3\_), p4(p4\_) {}

void get\_coordinates(const Point& p1, const Point& p2, const Point& p3, Point const& p4) {

this->p1 = p1;

this->p2 = p2;

this->p3 = p3;

this->p4 = p4;

}

Point center() {

return Point((this->p1.x + this->p2.x + this->p3.x + this->p4.x) / 4, (this->p1.y + this->p2.y + this->p3.y + this->p4.y) / 4);

}

double area() {

double a = distance(this->p1, this->p2);

double b = distance(this->p2, this->p3);

double c = distance(this->p3, this->p4);

double d = distance(this->p4, this->p1);

double p = (a + b + c + d) / 2;

return sqrt(abs(p - a) \* abs(p - b) \* abs(p - c) \* abs(p - d));

}

void print\_figure() const {

cout << '\n';

cout << "Rectangle:" << '\n';

cout << "First point: (" << this->p1.x << "," << this->p1.y << ")" << '\n';

cout << "Second point: (" << this->p2.x << "," << this->p2.y << ")" << '\n';

cout << "Third point: (" << this->p3.x << "," << this->p3.y << ")" << '\n';

cout << "Fourth point: (" << this->p4.x << "," << this->p4.y << ")" << '\n';

}

};

istream& operator>> (istream& is, Square& sqr) {

double x1, y1, x2, y2;

if (!(is >> x1 >> y1 >> x2 >> y2)) {

print\_error();

abort;

}

Point p1(x1, y1); Point p2(x2, y2);

sqr.get\_point1(p1); sqr.get\_point2(p2);

return is;

}

istream& operator>> (istream& is, Triangle& tri) {

double x1, y1, x2, y2, x3, y3;

if (!(is >> x1 >> y1 >> x2 >> y2 >> x3 >> y3)) {

print\_error();

abort;

}

Point p1(x1, y1); Point p2(x2, y2); Point p3(x3, y3);

tri.get\_coordinates(p1, p2, p3);

return is;

}

istream& operator>> (istream& is, Rectangle& rec) {

double x1, y1, x2, y2, x3, y3, x4, y4;

if (!(is >> x1 >> y1 >> x2 >> y2 >> x3 >> y3 >> x4 >> y4)) {

print\_error();

abort;

}

Point p1(x1, y1); Point p2(x2, y2); Point p3(x3, y3); Point p4(x4, y4);

if ((distance(p1, p2) != distance(p3, p4)) || (distance(p2, p3) != distance(p4, p1))) {

print\_error();

abort;

}

rec.get\_coordinates(p1, p2, p3, p4);

return is;

}

double area\_in\_vector(std::vector<Figure\*> figures) { // Сумма площадей фигур в векторе

double sum = 0;

vector<Figure\*>::iterator iter = figures.begin();

while (iter != figures.end()) {

sum += (\*iter)->area();

iter++;

}

return sum;

}

void delete\_el(vector<Figure\*>& figures, int i = 0) {

figures.erase(figures.begin() + i);

}

int main() {

setlocale(LC\_ALL, "rus");

vector<Figure\*> figures;

cout << "Введите 4 точки прямоугольника в порядке обхода в формате x1 y1: " << '\n';

Rectangle r;

if (!(cin >> r)) {

return 1;

}

cout << "Введите 2 точки диагонали квадрата в формате x1 y1: " << '\n';

Square s;

if (!(cin >> s)) {

return 1;

}

cout << "Введите 3 точки треугольника в порядке обхода в формате x1 y1: " << '\n';

Triangle t;

if (!(cin >> t)) {

return 1;

}

if (r.area() <= 0 || s.area() <= 0 || t.area() <= 0) { // проверка на корректность фигуры

print\_error();

return 1;

}

figures.push\_back(&r);

figures.push\_back(&t);

figures.push\_back(&s);

cout << "Фигуры в векторе: " << '\n';

vector<Figure\*>::iterator iter = figures.begin();

while (iter != figures.end()) {

(\*iter)->print\_figure();

cout << "Площадь = " << (\*iter)->area() << '\n';

cout << "Центр x = " << (\*iter)->center().x << ", y = " << (\*iter)->center().y << '\n';

++iter;

}

cout << "\n\nСумма всех площадей в векторе: " << area\_in\_vector(figures) << '\n';

cout << "Удаляем элемент с индексом 0" << '\n';

//удаляем элемент с индексом 0

delete\_el(figures, 0);

iter = figures.begin();

while (iter != figures.end()) {

(\*iter)->print\_figure();

cout << "Площадь = " << (\*iter)->area() << '\n';

cout << "Центр x = " << (\*iter)->center().x << ", y = " << (\*iter)->center().y << '\n';

++iter;

}

cout << "Сумма всех площадей в векторе после удаления первого элемента: " << area\_in\_vector(figures) << '\n';

return 0;

}

**ЛИТЕРАТУРА**

1. Формула Герона [Электронный ресурс].

URL: https://ru.wikipedia.org/wiki/Формула\_Герона (дата обращения: 22.10.2021).