МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ  
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСТИТЕТ)

**ЛАБОРАТОРНАЯ РАБОТА №3**

по курсу “Объектно-ориентированное программирование”

I семестр, 2021/22 учебный год

Студент: *Степанов Данила Михайлович, группа М8О-207Б-20*

Преподаватель: *Дорохов Евгений Павлович, каф. 806*

**Задание:** Спроектировать и запрограммировать на языке C++ классы трёх фигур. Классы должны удовлетворять следующим правилам:

* + Должны быть названы как в вариантах задания и расположены в раздельных файлах;
  + Иметь общий родительский класс Figure;
  + Содержать конструктор по умолчанию;
  + Содержать конструктор, принимающий координаты вершин фигуры из стандартного потока std::cin, расположенных через пробел (например: 0.0 0.0 1.0 0.0 1.0 1.0 0.0 1.0);
  + Содержать набор общих методов:
    - size\_t VertexesNumber() – метод, возвращающий количество вершин фигуры
    - double Area() – метод расчета площади фигуры
    - dvoid Print(std::ostream& os) – метод печати типа фигуры и ее координат вершин в поток вывода os в формате:

Rectangle: (0.0, 0.0) (1.0, 0.0) (1.0, 1.0) (0.0, 1.0)\n

**Вариант №24:**

* + Фигура 1: 8-угольник (Octagon)
  + Фигура 2: Треугольник (Triangle)
  + Фигура 3: Квадрат (Square)

**Описание программы:**

Исходный код разделён на 10 файлов:

* point.h – описание класса точки
* point.cpp – реализация класса точки
* figure.h – описание класса фигуры
* octagon.h – описание класса прямоугольника (наследуется от фигуры)
* octagon.cpp – реализация класса прямоугольника
* triangle.h – описание класса квадрата (наследуется от прямоугольника)
* triangle.cpp – реализация класса квадрата
* square.h – описание класса трапеции (наследуется от фигуры)
* square.cpp – реализация класса трапеции
* main.cpp – основная программа

**Дневник отладки:**

Проблем не возникло.

**Тестирование программы:**

Enter the points' coordinates of triangle:

1 3 2 4 5 8

Triangle's number of vertexes: 3

Triangles's area: 0.5

Figure type:

Triangle: (1, 3) (2, 4) (5, 8)

Enter the points' coordinates of square:

1 1 2 2 3 3 4 4

Square's number of vertexes: 4

Square's area: 1

Figure type:

Square: (1, 1) (2, 2) (3, 3) (4, 4)

Enter the points' coordinates of octagon:

1 4 1 2 5 6 2 8 3 1 2 6 9 5 5 4

Octagon's number of vertexes: 8

Octagon's area: 0.5

Figure type:

Octagon: (1, 4) (1, 2) (5, 6) (2, 8) (3, 1) (2, 6) (9, 5) (5, 4)

**Вывод:**  
 В лабораторной работе я, создавая классы трёх фигур с соответствии с вариантом задания, познакомился с таким понятиями как наследование, дружественные функции, перегрузка операторов и с операциями ввода-вывода из стандартных библиотек.

**Исходный код:**

**point.h:**

#ifndef POINT\_H

#define POINT\_H

#include <iostream>

class Point {

public:

Point();

Point(std::istream &is);

Point(double x, double y);

double dist(Point& other);

double getX();

double getY();

friend std::istream& operator>>(std::istream& is, Point& p);

friend std::ostream& operator<<(std::ostream& os, Point& p);

private:

double x\_;

double y\_;

};

#endif

**point.cpp:**

#include "point.h"

#include <cmath>

Point::Point() : x\_(0.0), y\_(0.0) {}

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

Point::Point(std::istream &is) {

is >> x\_ >> y\_;

}

double Point::dist(Point& other) {

double dx = (other.x\_ - x\_);

double dy = (other.y\_ - y\_);

return std::sqrt(dx\*dx + dy\*dy);

}

double Point::getX()

{

return x\_;

}

double Point::getY()

{

return y\_;

}

std::istream& operator>>(std::istream& is, Point& p) {

is >> p.x\_ >> p.y\_;

return is;

}

std::ostream& operator<<(std::ostream& os, Point& p) {

os << "(" << p.x\_ << ", " << p.y\_ << ")";

return os;

}

**figure.h:**

#ifndef FIGURE\_H

#define FIGURE\_H

#include "point.h"

class Figure

{

public:

virtual size\_t VertexesNumber() = 0;

virtual double Area() = 0;

virtual void Print(std::ostream& os) = 0;

};

#endif

**octagon.h:**

#ifndef OCTAGON\_H

#define OCTAGON\_H

#include "figure.h"

class Octagon : Figure

{

public:

Octagon(std::istream& is);

size\_t VertexesNumber();

double Area();

void Print(std::ostream& os);

private:

Point a\_, b\_, c\_, d\_;

Point e\_, f\_, g\_, h\_;

};

#endif

**octagon.cpp:**

#include "octagon.h"

Octagon::Octagon(std::istream& is)

{

std::cin >> a\_ >> b\_ >> c\_ >> d\_;

std::cin >> e\_ >> f\_ >> g\_ >> h\_;

}

size\_t Octagon::VertexesNumber()

{

return (size\_t)8;

}

double Octagon::Area()

{

return 0.5 \* abs((a\_.getX() \* b\_.getY() + b\_.getX() \* c\_.getY() + c\_.getX() \* d\_.getY() + d\_.getX() \* e\_.getY() + e\_.getX() \* f\_.getY() +

f\_.getX() \* g\_.getY() + g\_.getX() \* h\_.getY() + h\_.getX() \* a\_.getY() - (b\_.getX() \* a\_.getY() + c\_.getX() \* b\_.getY() +

d\_.getX() \* c\_.getY() + e\_.getX() \* d\_.getY() + f\_.getX() \* e\_.getY() + g\_.getX() \* f\_.getY() + h\_.getX() \* g\_.getY() +

a\_.getX() \* h\_.getY())));

}

void Octagon::Print(std::ostream& os)

{

std::cout << "Octagon: " << a\_ << " " << b\_ << " ";

std::cout << c\_ << " " << d\_ << " " << e\_ << " ";

std::cout << f\_ << " " << g\_ << " " << h\_ << "\n";

}

**triangle.h:**

#ifndef TRIANGLE\_H

#define TRIANGLE\_H

#include "figure.h"

class Triangle : Figure

{

public:

Triangle(std::istream&);

size\_t VertexesNumber();

double Area();

void Print(std::ostream& os);

private:

Point a\_;

Point b\_;

Point c\_;

};

#endif

**triangle.cpp:**

#include "triangle.h"

#include "figure.h"

#include <math.h>

Triangle::Triangle(std::istream& is)

{

std::cin >> a\_ >> b\_ >> c\_;

}

size\_t Triangle::VertexesNumber()

{

return (size\_t)3;

}

double Triangle::Area()

{

double a = a\_.dist(b\_);

double b = b\_.dist(c\_);

double c = a\_.dist(c\_);

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

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

}

void Triangle::Print(std::ostream& os)

{

std::cout << "Triangle: " << a\_ << " " << b\_ << " " << c\_ << "\n";

}

**square.h:**

#ifndef SQUARE\_H

#define SQUARE\_H

#include "figure.h"

class Square : Figure

{

public:

Square(std::istream& is);

size\_t VertexesNumber();

double Area();

void Print(std::ostream& os);

private:

Point a\_;

Point b\_;

Point c\_;

Point d\_;

};

#endif

**main.cpp:**

#include "figure.h"

#include "triangle.h"

#include "square.h"

#include "octagon.h"

int main()

{

std::cout << "Enter the points' coordinates of triangle:\n";

triangle a(std::cin);

std::cout << "Triangle's number of vertexes: " << a.VertexesNumber() << "\n";

std::cout << "Triangles's area: " << a.Area() << "\n";

std::cout << "Figure type:\n";

a.Print(std::cout);

std::cout << "Enter the points' coordinates of square:\n";

square b(std::cin);

std::cout << "Square's number of vertexes: " << b.VertexesNumber() << "\n";

std::cout << "Square's area: " << b.Area() << "\n";

std::cout << "Figure type:\n";

b.Print(std::cout);

std::cout << "Enter the points' coordinates of octagon:\n";

octagon c(std::cin);

std::cout << "Octagon's number of vertexes: " << c.VertexesNumber() << "\n";

std::cout << "Octagon's area: " << c.Area() << "\n";

std::cout << "Figure type:\n";

c.Print(std::cout);

return 0;

}

**square.cpp:**

#include "square.h"

Square::Square(std::istream& is)

{

std::cin >> a\_ >> b\_ >> c\_ >> d\_;

}

size\_t Square::VertexesNumber()

{

return (size\_t)4;

}

double Square::Area()

{

int a = a\_.dist(b\_);

return a \* a;

}

void Square::Print(std::ostream& os)

{

std::cout << "Square: " << a\_ << " " << b\_ << " " << c\_ << " " << d\_ << "\n";

}