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";
}
```