**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ**
**(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСТИТЕТ)**

# ЛАБОРАТОРНАЯ РАБОТА №1

**по курсу объектно-ориентированное программирование III семестр, 2021/22 уч. год**

Студент *Тихонов Фёдор Андреевич, группа М8О-207Б-20*
Преподаватель *Дорохов Евгений Павлович*

## Условие

Задание: Вариант 26: Квадрат, Прямоугольник, Трапеция. Необходимо спроектировать и запрограммировать на языке C++ классы трех фигур, согласно варианту задания. Классы должны удовлетворять следующим правилам:

1. Должны быть названы также, как в вариантах задания и расположенны в раздельных файлах: отдельно заголовки (имя_класса_с_маленькой_буквы.h), отдельно описание методов (имя_класса_с_маленькой_буквы.cpp).
2. Иметь общий родительский класс Figure;
3. Содержать конструктор, принимающий координаты вершин фигуры из стандартного потока std::cin, расположенных через пробел. Пример: "0.0 0.0 1.0 0.0 1.0 1.0 0.0 1.0"
4. Содержать набор общих методов:
   - size_t VertexesNumber() - метод, возвращающий количество вершин фигуры;
   - double Area() - метод расчета площади фигуры;
   - void 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)"с переводом строки в конце.

## Описание программы

Исходный код лежит в 11 файлах:

1. main.cpp: основная программа, взаимодействие с пользователем посредством команд из меню
2. figure.h: описание абстрактного класса фигур
3. point.h: описание класса точки
4. square.h: описание класса квадрата, наследующегося от rectangle
5. rectangle.h: описание класса прямоугольника, наследующегося от figures
6. trapezoid.h: описание класса трапеции, наследующегося от figures
7. point.cpp: реализация класса точки
8. square.cpp: реализация класса квадрата, наследующегося от rectangle
9. rectangle.cpp: реализация класса прямоугольника, наследующегося от figures
10. trapezoid.cpp: реализация класса трапеции, наследующегося от figures

## Выводы

В данной лабораторной работе я познакомился с принципами и концпециями объектно-ориентированного программирования: инкапсуляцией, наследованием и полиморфизмом. Научился проектировать классы и работать с ними, а также поработал с конструкторами, деструкторами и виртуальными функциями в C++.

## Исходный код

### figure.h

```
#ifndef FIGURE_H
#define FIGURE_H

#include <iostream>
#include "point.h"

class Figure {
public:
    virtual size_t VertexesNumber() = 0;
    virtual double Area() = 0;
    virtual void Print(std::ostream& os) = 0;
    ~Figure() {};
};

#endif //FIGURE_H
```

# point.h

```
#ifndef POINT_H
#define POINT_H

#include <iostream>

class Point {
public:
    Point();
    Point(std::istream &is);
    Point(double x, double y);
    double fx();
    double fy();
    double dist(Point& other);
    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_H
```

# point.cpp

```cpp
#include <iostream>
#include <cmath>
#include "point.h"

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::fx(){
    return x_;
};

double Point::fy(){
    return y_;
};

double Point::dist(Point& other) {
    double dx = (other.x_ - x_);
    double dy = (other.y_ - y_);
    return std::sqrt(dx*dx + dy*dy);
}

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;
}
```

## main.cpp

```cpp
#include <iostream>
#include "point.h"
#include "figure.h"
#include "square.h"
#include "rectangle.h"
#include "trapezoid.h"

int main() {
    std::cout << "Enter a coordinates of \"Square\"" << std::endl;
    Square a(std::cin);
    a.Print(std::cout);
    std::cout << a.Area() << "\n";

    std::cout << "Enter a coordinates of \"Rectangle\"" << std::endl;
    Rectangle b(std::cin);
    b.Print(std::cout);
    std::cout << b.Area() << "\n";

    std::cout << "Enter a coordinates of \"Trapezoid\"" << std::endl;
    Trapezoid c(std::cin);
    c.Print(std::cout);
    std::cout << c.Area() << std::endl;
}
```