

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

Лабораторная работа №3

по курсу “Объектно-ориентированное программирование”

I семестр, 2021/22 учебный год

Студент: Капичников Ярослав Андреевич, 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 отдельных файлах;
- Иметь общий родительский класс Figure;
- 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;
 - `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).

Вариант №10:

- Фигура 1: Прямоугольник (Rectangle)
- Фигура 2: Трапеция (Trapezoid)
- Фигура 3: Ромб (Rhombus)

Описание программы:

Исходный код разделён на 10 файлов:

- `point.h` – описание класса точки
- `point.cpp` – реализация класса точки
- `figure.h` – описание класса фигуры
- `rectangle.h` – описание класса прямоугольника (наследуется от фигуры)
- `rectangle.cpp` – реализация класса прямоугольника
- `rhombus.h` – описание класса ромба (наследуется от прямоугольника)
- `rhombus.cpp` – реализация класса ромба
- `trapezoid.h` – описание класса трапеции (наследуется от фигуры)
- `trapezoid.cpp` – реализация класса трапеции
- `main.cpp` – основная программа

Дневник отладки

№	Дата	Событие	Действие по исправлению
1	10.10.21	Обнаружил, что все мои фигуры - четырехугольники	Добавил в класс Figure 4 точки

Вывод:

Проделав данную работу, я изучил базовые понятия ооп, познакомился с классами и научился создавать дочерние подклассы. Также использовал перегрузку операторов ввода и вывода. Эти знания пригодятся в будущем любому программисту.

Исходный код:

Figure.h

```
#pragma
once

#include <iostream>
#include "point.h"
using namespace std;
class Figure {
public:
    virtual size_t VertexesNumber() = 0;
    virtual double Area() = 0;
    virtual void Print(std::ostream& os) = 0;
protected:
    Point a;
    Point b;
    Point c;
    Point d;
};
```

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;
    }
}

```

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

```

Rectangle.cpp

```

#include
<iostream>
#include"point.h"
#include"rectangle.h"
using namespace std;
Rectangle::Rectangle(Point a1, Point a2, Point a3, Point a4) {
    a = a1;
    b = a2;
    c = a3;
    d = a4;
}
double Rectangle::Area() {
    double A = a.dist(b);
    double B = b.dist(c);
    return A * B;
}
void Rectangle::Print(std::ostream& os)
{
    std::cout << "Rectangle: " << a << " " << b << " " << c << " " << d << endl;
}
size_t Rectangle::VertexesNumber()
{
    return (size_t)4;
}
Rectangle::Rectangle(std::istream& is) {

```

```

        cin >> a >> b >> c >> d;
    }

```

Rectangle.h

```

#include
<iostream>

#include"point.h"
#include"figure.h"
class Rectangle : Figure {
public:
    double Area();
    void Print(std::ostream& os);
    size_t VertexesNumber();
    Rectangle(Point a1, Point a2, Point a3, Point a4);
    Rectangle(std::istream& is);
};

```

Rhombus.cpp

s (26 sloc) 539 Bytes

[RawBlame](#)

```

#include <iostream>
#include"point.h"
#include"rhombus.h"
using namespace std;
Rhombus::Rhombus(Point a1, Point a2, Point a3, Point a4) {
    a = a1;
    b = a2;
    c = a3;
    d = a4;
}
double Rhombus::Area() {
    double A = a.dist(c);
    double B = b.dist(d);
    return A * B / 2;
}
void Rhombus::Print(std::ostream& os)
{
    std::cout << "Rhombus: " << a << " " << b << " " << c << " " << d << endl;
}
size_t Rhombus::VertexesNumber()
{
    return (size_t)4;
}
Rhombus::Rhombus(std::istream& is) {
    cin >> a >> b >> c >> d;
}

```

Rhombus.h

```

#include
<iostream>

#include"point.h"
#include"figure.h"
class Rhombus : Figure {
public:
    double Area();
    void Print(std::ostream& os);
    size_t VertexesNumber();
    Rhombus(Point a1, Point a2, Point a3, Point a4);
    Rhombus(std::istream& is);
};

```

Trapezoid.cpp

```
#include
<iostream>

#include "point.h"
#include "trapezoid.h"
#include <math.h>
using namespace std;
Trapezoid::Trapezoid(Point a1, Point a2, Point a3, Point a4) {
    a = a1;
    b = a2;
    c = a3;
    d = a4;
}
double Trapezoid::Area() {

    double la = a.dist(d);
    double lb = b.dist(c);
    double lc = c.dist(d);
    double ld = a.dist(b);
    if (la > lb) {
        double t = la;
        la = lb;
        lb = t;
    }
    double numerator = (lb - la) * (lb - la) + lc * lc - ld * ld;
    double denominator = 2 * (lb - la);
    if (denominator == 0) {
        return (la * lc);
    }
    double h = sqrt(lc * lc - ((numerator * numerator) / (denominator *
denominator)));
    return ((la + lb) / 2 * h);

}
void Trapezoid::Print(std::ostream& os)
{
    std::cout << "Trapezoid: " << a << " " << b << " " << c << " " << d << endl;
}
size_t Trapezoid::VertexesNumber()
{
    return (size_t)4;
}
Trapezoid::Trapezoid(std::istream& is) {
    cin >> a >> b >> c >> d;
}
```

Trapezoid.h

```
#include
<iostream>

#include "point.h"
#include "figure.h"
class Trapezoid : Figure {
public:
    double Area();
    void Print(std::ostream& os);
    size_t VertexesNumber();
    Trapezoid(Point a1, Point a2, Point a3, Point a4);
    Trapezoid(std::istream& is);
};
```

