МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСТИТЕТ)

ЛАБОРАТОРНАЯ РАБОТА №1

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

Студент *Пономарев Никита Владимирович, группа М8О-207Б-20*

Преподаватель *Дорохов Евгений Павлович*

Условие

Задание: Вариант 19: Прямоугольник, трапеция, ромб. 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. src/main.cpp: основная программа, взаимодействие с пользователем посредством комманд из меню
2. include/figure.h: описание абстрактного класса фигур
3. include/point.h: описание класса точки
4. include/rectangle.h: описание класса прямоугольника, наследующегося от figures
5. include/rhombus.h: описание класса ромба, наследующегося от figures
6. include/trapezoid.h: описание класса трапеции, наследующегося от figure
7. include/point.cpp: реализация класса точки
8. include/rectangle.cpp: реализация класса прямоугольника, наследующегося от figures
9. include/rhombus.cpp: реализация класса ромба, наследующегося от figures
10. include/trapezoid.cpp: реализация класса трапеции, наследующегося от figure

Дневник отладки

Недочёты

Выводы

В процессе выполнения работы я на практике познакомился с принципами ООП, реализовал несколько классов данных(фигуры), и для каждого из них - функции. Научился перегружать операторы для более комфортной работы с моими классами.

Исходный код:

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;
        ~Figure() {
            std::cout << "Delete succesfully!\n";
        };
};

#endif
```

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 X();
        double Y();
        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

```
#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::X(){
  return x_;
};

double Point::Y(){
  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;
}
```

rectangle.h

```
#ifndef RECTANGLE_H
#define RECTANGLE_H
```

```
#include "figure.h"

class Rectangle: Figure {
    public:
        size_t VertexesNumber();
        double Area();
        void Print(std::ostream& os);
        Rectangle();
        Rectangle(Point a_, Point b_, Point c_, Point d_);
        Rectangle(std::istream& is);
    private:
        Point a;
        Point b;
        Point c;
        Point d;
};

#endif
```

rectangle.cpp

```
#include "point.h"
#include "rectangle.h"

double Rectangle::Area(){
    return a.dist(b) * b.dist(c);
}

void Rectangle::Print(std::ostream& os){
    os << a << " " << b << " " << c << " " << d << "\n";
}

size_t Rectangle::VertexesNumber(){
    return (size_t)(4);
}

Rectangle::Rectangle() : a(Point()), b(Point()), c(Point()), d(Point()){
}

Rectangle::Rectangle(Point a_, Point b_, Point c_, Point d_):
                                a(a_), b(b_), c(c_), d(d_){
}
```

```
Rectangle::Rectangle(std::istream& is){
    is >> a >> b >> c >> d;
}
```

rhombus.h

```
#ifndef RHOMBUS_H
#define RHOMBUS_H

#include "figure.h"

class Rhombus: Figure {
    public:
        void Print(std::ostream& os);
        double Area();
        size_t VertexesNumber();
        Rhombus();
        Rhombus(Point a_, Point b_, Point c_, Point d_);
        Rhombus(std::istream& is);
    private:
        Point a;
        Point b;
        Point c;
        Point d;
};

#endif
```

rhombus.cpp

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

Rhombus::Rhombus() : a(Point()), b(Point()), c(Point()), d(Point()) {
}

Rhombus::Rhombus(Point a_, Point b_, Point c_, Point d_) :
                        a(a_), b(b_), c(c_), d(d_) {
}

Rhombus::Rhombus(std::istream& is){
```

```
    is >> a >> b >> c >> d;
}

void Rhombus::Print(std::ostream& os){
    os << a << " " << b << " " << c << " " << d << "\n";
}

size_t Rhombus::VertexesNumber(){
    return (size_t)(4);
}

double Rhombus::Area(){
    return a.dist(c) * b.dist(d) * 0.5;
}
```

trapezoid.h

```
#ifndef TRAPEZOID_H
#define TRAPEZOID_H

#include "figure.h"

class Trapezoid: Figure {
    public:
        void Print(std::ostream& os);
        double Area();
        size_t VertexesNumber();
        Trapezoid();
        Trapezoid(Point a_, Point b_, Point c_, Point d_);
        Trapezoid(std::istream& is);
    private:
        Point a;
        Point b;
        Point c;
        Point d;
};

#endif
```

trapezoid.cpp

```
#include "figure.h"
#include "trapezoid.h"
```

```cpp
#include <cmath>

Trapezoid::Trapezoid(): a(Point()), b(Point()), c(Point()), d(Point()) {
}

Trapezoid::Trapezoid(Point a_, Point b_, Point c_, Point d_):
                            a(a_), b(b_), c(c_), d(d_) {
}

Trapezoid::Trapezoid(std::istream& is) {
    is >> a >> b >> c >> d;
}

void Trapezoid::Print(std::ostream& os) {
    os << a << " " << b << " " << c << " " << d << "\n";
}

size_t Trapezoid::VertexesNumber() {
    return (size_t)(4);
}

double Trapezoid::Area(){
    double ax = a.X() - c.X();
    double bx = b.X() - d.X();
    double ay = a.Y() - c.Y();
    double by = b.Y() - d.Y();
    double COS = (ax*bx + ay*by)/(sqrt(ax*ax+ay*ay)*sqrt(bx*bx+by*by));
    return double(a.dist(c) * b.dist(d) * 0.5 *sin(acos(COS)));
}
```

main.cpp

```cpp
#include "point.h"
#include "rectangle.h"
#include "rhombus.h"
#include "trapezoid.h"
#include "figure.h"

int main(){
    std::cout << "Please, enter coordinates of Rectangle\n";
    Rectangle a(std::cin);
    a.Print(std::cout);
    std::cout << a.Area() << "\n";
```

```
    std::cout << "Please, enter coordinates of Trapezoid\n";
    Trapezoid b(std::cin);
    b.Print(std::cout);
    std::cout << b.Area() << "\n";
    std::cout << "Please, enter coordinates of Rhombus\n";
    Rhombus c(std::cin);
    c.Print(std::cout);
    std::cout << c.Area() << "\n";
}
```