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: *Пономарев Никита Владимирович, 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, 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

Вариант №19:

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

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 ромб (наследуется от а)
- `rhombus.cpp` – реализация класса ромб
- `trapezoid.h` – описание класса трапеции (наследуется от фигуры)
- `trapezoid.cpp` – реализация класса трапеции
- `main.cpp` – основная программа

Дневник отладки:

Возникли проблемы при вычислении площади трапеции. В моей программе была использована формула, вычисляющая площадь как произведение диагоналей на половину косинуса угла между ними. Из за неправильной методики находения косинуса, площадь вычислялась неверно. Этот недочет удалось заметить при тестировании и, изменив формулу рассчета косинуса на более общую, и исправить.

Вывод:

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:

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

#endif
```

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);
        friend std::istream &operator>>(std::istream &is, Trapezoid &figure);
        friend std::ostream &operator<<(std::ostream &os, const Trapezoid &figure);
    private:
        Point a;
        Point b;
        Point c;
        Point d;
};

#endif
```

trapezoid.cpp:

```
#include "figure.h"
#include "trapezoid.h"
#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)));
}

std::istream &operator>>(std::istream &is, Trapezoid &figure){
    is >> figure.a >> figure.b >> figure.c >> figure.d;
    return is;
}

std::ostream &operator<<(std::ostream &os, const Trapezoid &figure){
    os << "Trapezoid: " << figure.a << " " << figure.b << " " << figure.c << " " << figure.d << std::endl;
    return os;
}
```

rectangle.h:

```
#ifndef RECTANGLE_H
#define RECTANGLE_H

#include "figure.h"

class Rectangle: public 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);
        friend std::istream &operator>>(std::istream &is, Rectangle &figure);
        friend std::ostream &operator<<(std::ostream &os, const Rectangle &figure);
    private:
        Point a;
        Point b;
        Point c;
        Point d;
};

#endif
std::ostream &operator<<(std::ostream &os, const Rectangle &figure){
    os << "Rectangle: " << figure.a << " " << figure.b << " " << figure.c << " " << figure.d << std::endl;
    return os;
}
```

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;
}

std::istream &operator>>(std::istream &is, Rectangle &figure){
    is >> figure.a >> figure.b >> figure.c >> figure.d;
    return is;
}

std::ostream &operator<<(std::ostream &os, const Rectangle &figure){
    os << "Rectangle: " << figure.a << " " << figure.b << " " << figure.c << " " << figure.d << std::endl;
    return os;
}
```

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);
```

```
        friend std::istream &operator>>(std::istream &is, Rhombus &figure);
        friend std::ostream &operator<<(std::ostream &os, const Rhombus &figure);
    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;
}

std::istream &operator>>(std::istream &is, Rhombus &figure){
    is >> figure.a >> figure.b >> figure.c >> figure.d;
    return is;
}

std::ostream &operator<<(std::ostream &os, const Rhombus &figure){
    os << "Rhombus: " << figure.a << " " << figure.b << " " << figure.c << " " << figure.d << std::endl;
    return os;
}
```

main.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";
```

```
}
```

Пример работы:

```
Please, enter coordinates of Rectangle
1 2 3 4
5 6
7 8
(1, 2) (3, 4) (5, 6) (7, 8)
8
Please, enter coordinates of Trapezoid
2 0
0 0
1 1
2 1
(2, 0) (0, 0) (1, 1) (2, 1)
1.5
Please, enter coordinates of Rhombus
0 0
1 1
2 2
3 3
(0, 0) (1, 1) (2, 2) (3, 3)
4
Delete succesfully!
Delete succesfully!
Delete succesfully!
```