

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

ЛАБОРАТОРНАЯ РАБОТА №3

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

Студент Фаттахетдинов Сильвестр Динарович, группа М8О-208Б-20

Преподаватель Дорохов Евгений Павлович

Условие

Задание: Вариант 21: Ромб, Пятиугольник, Шестиугольник. 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 расположены в отдельных 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 команд из меню
2. include/figure.h: описание абстрактного класса фигур
3. include/point.h: описание класса точки
4. include/rhombus.h: описание класса ромба, наследующегося от figures
5. include/pentagon.h: описание класса пятиугольника, наследующегося от figures
6. include/hexagon.h: описание класса шестиугольника, наследующегося от figures
7. include/point.cpp: реализация класса точки
8. include/rhombus.cpp: реализация класса ромба, наследующегося от figures
9. include/pentagon.cpp: реализация класса пятиугольника, наследующегося от figures
10. include/hexagon.cpp: реализация класса шестиугольника, наследующегося от figures

Дневник отладки

Недочёты

Выводы

Я научился реализовывать классы в C++, а также познакомился с дружественными функциями, перегрузкой операторов и изучил основные понятия и принципы ООП.

Исходный код

figure.h

```
#ifndef FIGURE_H
#define FIGURE_H
#include <cstddef>

#include "point.h"
class Figure {
public:
    virtual size_t VertexesNumber() = 0;
    virtual double Area() = 0;
    virtual void Print(std::ostream& os) = 0;
    virtual ~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 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 <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::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;
}
```

rhombus.h

```
#ifndef RHOMBUS_H
#define RHOMBUS_H

#include <iostream>

#include "figure.h"

class Rhombus : public Figure {
public:
    Rhombus();
    Rhombus(std::istream& is);
    Rhombus(Point a, Point b, Point c, Point d);
    Rhombus(const Rhombus& other);
    virtual ~Rhombus();
    size_t VertexesNumber();
    double Area();
    void Print(std::ostream& os);

private:
    Point a, b, c, d;
};

#endif // RHOMBUS_H
```

rhombus.cpp

```
#include <cmath>

#include "rhombus.h"
Rhombus::Rhombus() : a(0.0, 0.0), b(0.0, 0.0), c(0.0, 0.0), d(0.0, 0.0) {
    std::cout << "Default Rhombus created" << std::endl;
}
Rhombus::Rhombus(std::istream& is) {
    is >> a >> b >> c >> d;
}
Rhombus::Rhombus(Point _a, Point _b, Point _c, Point _d) {
    if (sqrt((_b.x() - _a.x()) * (_b.x() - _a.x()) +
            (_b.y() - _a.y()) * (_b.y() - _a.y()))) ==
        sqrt((_c.x() - _b.x()) * (_c.x() - _b.x()) +
            (_c.y() - _b.y()) * (_c.y() - _b.y()))) &&
```

```

        sqrt((_c.x() - _b.x()) * (_c.x() - _b.x()) +
            (_c.y() - _b.y()) * (_c.y() - _b.y())) ==
        sqrt((_d.x() - _c.x()) * (_d.x() - _c.x()) +
            (_d.y() - _c.y()) * (_d.y() - _c.y())) &&
        sqrt((_d.x() - _c.x()) * (_d.x() - _c.x()) +
            (_d.y() - _c.y()) * (_d.y() - _c.y())) ==
        sqrt((_a.x() - _d.x()) * (_a.x() - _d.x()) +
            (_a.y() - _d.y()) * (_a.y() - _d.y())) {
    a = _a;
    b = _b;
    c = _c;
    d = _d;
} else {
    std::cout << "Invalid arguments";
}
}
Rhombus::Rhombus(const Rhombus& other)
    : Rhombus(other.a, other.b, other.c, other.d) {}
void Rhombus::Print(std::ostream& os) {
    os << "Rhombus:";
    os << a << b << c << d << std::endl;
    // os << "(" << b << "." << b << ")" ";
    // os << "(" << c << "." << c << ")" ";
    // os << "(" << d << "." << d << ")" ";
}
double Rhombus::Area() {
    double s =
        abs(a.x() * b.y() + b.x() * c.y() + c.x() * d.y() + d.x() * a.y() -
            b.x() * a.y() - c.x() * b.y() - d.x() * c.y() - a.x() * d.y()) /
        2;

    return s;
}
size_t Rhombus::VertexesNumber() {
    return 4;
}
Rhombus::~Rhombus() {
    std::cout << "Rhombus deleted" << std::endl;
}

```

pentagon.h


```

#ifndef PENTAGON_H
#define PENTAGON_H

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

class Pentagon : public Figure {
public:
    Pentagon();
    Pentagon(std::istream& is);
    Pentagon(Point a, Point b, Point c, Point d, Point e);
    Pentagon(const Pentagon& other);
    virtual ~Pentagon();
    size_t VertexesNumber();
    double Area();
    void Print(std::ostream &os);
private:
    Point a,b,c,d,e;
};

#endif // PENTAGON_H

```

pentagon.cpp

```

#include "pentagon.h"
#include <cmath>

Pentagon::Pentagon()
    : a(0.0, 0.0), b(0.0, 0.0), c(0.0, 0.0), d(0.0, 0.0), e(0.0, 0.0) {
    std::cout << "Default Pentagon created" << std::endl;
}

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

Pentagon::Pentagon(Point _a, Point _b, Point _c, Point _d, Point _e) {
    a = _a;
    b = _b;
    c = _c;
    d = _d;
    e = _e;
}

Pentagon::Pentagon(const Pentagon& other) : Pentagon(other.a, other.b, other.c,
other.d, other.e) {
}

```

```

void Pentagon::Print(std::ostream &os){
    os << "Pentagon:";
    os << a << b << c << d << e << std::endl;
    // os << "(" << b << "." << b << ")" ";
    // os << "(" << c << "." << c << ")" ";
    // os << "(" << d << "." << d << ")" ";
}
double Pentagon::Area(){
    double s = abs(a.x() * b.y() + b.x() * c.y() + c.x() * d.y() + d.x() * e.y() +
e.x() * a.y() - b.x() * a.y() - c.x() * b.y() - d.x() * c.y() - e.x() * d.y() - a.x() * e.y()
)/2;

    return s;
}
size_t Pentagon::VertexesNumber(){
    return 4;
}
Pentagon::~Pentagon() {
    std::cout << "Pentagon deleted" << std::endl;
}

```

hexagon.h

```

#ifndef HEXAGON_H
#define HEXAGON_H

#include <iostream>

#include "figure.h"

class Hexagon : public Figure {
public:
    Hexagon();
    Hexagon(std::istream& is);
    Hexagon(Point a, Point b, Point c, Point d, Point e, Point f);
    Hexagon(const Hexagon& other);
    virtual ~Hexagon();
    size_t VertexesNumber();
    double Area();
    void Print(std::ostream& os);

private:

```

```

    Point a, b, c, d, e, f;
};

#endif // HEXAGON_H

```

hexagon.cpp

```

#include <cmath>

#include "hexagon.h"
Hexagon::Hexagon()
    : a(0.0, 0.0),
      b(0.0, 0.0),
      c(0.0, 0.0),
      d(0.0, 0.0),
      e(0.0, 0.0),
      f(0.0, 0.0) {
    std::cout << "Default Hexagon created" << std::endl;
}

Hexagon::Hexagon(std::istream& is) {
    is >> a >> b >> c >> d >> e >> f;
}

Hexagon::Hexagon(Point _a, Point _b, Point _c, Point _d, Point _e, Point _f) {
    a = _a;
    b = _b;
    c = _c;
    d = _d;
    e = _e;
    f = _f;
}

Hexagon::Hexagon(const Hexagon& other)
    : Hexagon(other.a, other.b, other.c, other.d, other.e, other.f) {}

void Hexagon::Print(std::ostream& os) {
    os << "Hexagon:";
    os << a << b << c << d << e << f << std::endl;
    // os << "(" << b << "." << b << ")" ";
    // os << "(" << c << "." << c << ")" ";
    // os << "(" << d << "." << d << ")" ";
}

double Hexagon::Area() {
    double s =
        abs(a.x() * b.y() + b.x() * c.y() + c.x() * d.y() + d.x() * e.y() +
            e.x() * f.y() + f.x() * a.y() - b.x() * a.y() - c.x() * b.y() -
            d.x() * c.y() - e.x() * d.y() - f.x() * e.y() - a.x() * f.y()) /

```

```

        2;

    return s;
}
size_t Hexagon::VertexesNumber() {
    return 4;
}
Hexagon::~~Hexagon() {
    std::cout << "Hexagon deleted" << std::endl;
}

```

main.cpp

```

#include <iostream>

#include "hexagon.h"
#include "pentagon.h"
#include "rhombus.h"

using namespace std;

int main() {
    std::cout << "Enter Rhombus coordinates" << std::endl;
    Rhombus a(std::cin);
    std::cout << a.VertexesNumber() << std::endl;
    a.Print(std::cout);
    std::cout << "Area is: " << a.Area() << std::endl;
    Rhombus a1;
    std::cout << "Default Rhombus coordinates: " << std::endl;
    a1.Print(std::cout);
    std::cout << "Default Rhombus area is: " << a1.Area() << std::endl;
    Rhombus a2(Point(0, 0), Point(0, 1), Point(1, 1), Point(1, 0));
    std::cout << "Rhombus 2 coordinates are: " << std::endl;
    a2.Print(std::cout);
    std::cout << "Rhombus 2 area is:" << a2.Area() << std::endl;
    Rhombus a3(a2);
    std::cout << "Rhombus 3 coordinates are: " << std::endl;
    a3.Print(std::cout);
    std::cout << "Rhombus 3 area is:" << a3.Area() << std::endl;

    std::cout << "Enter Pentagon coordinates" << std::endl;
    Pentagon b(std::cin);
    std::cout << b.VertexesNumber() << std::endl;
}

```

```

b.Print(std::cout);
std::cout << "Area is: " << b.Area() << std::endl;
Pentagon b1;
std::cout << "Default Pentagon coordinates: " << std::endl;
b1.Print(std::cout);
std::cout << "Default Pentagon area is: " << b1.Area() << std::endl;
Pentagon b2(Point(0, 0), Point(0, 1), Point(1, 1), Point(1.5, 0.5),
            Point(1, 0));
std::cout << "Pentagon 2 coordinates are: " << std::endl;
b2.Print(std::cout);
std::cout << "Pentagon 2 area is:" << b2.Area() << std::endl;
Pentagon b3(b2);
std::cout << "Pentagon 3 coordinates are: " << std::endl;
b3.Print(std::cout);
std::cout << "Pentagon 3 area is:" << b3.Area() << std::endl;

std::cout << "Enter Hexagon coordinates" << std::endl;
Hexagon c(std::cin);
std::cout << c.VertexesNumber() << std::endl;
c.Print(std::cout);
std::cout << "Area is: " << c.Area() << std::endl;
Hexagon c1;
std::cout << "Default Hexagon coordinates: " << std::endl;
c1.Print(std::cout);
std::cout << "Default Hexagon area is: " << c1.Area() << std::endl;
Hexagon c2(Point(0, 0), Point(0, 1), Point(1, 1), Point(1.5, 0.5),
            Point(1, 0.2), Point(1, 0));
std::cout << "Hexagon 2 coordinates are: " << std::endl;
c2.Print(std::cout);
std::cout << "Hexagon 2 area is:" << c2.Area() << std::endl;
Hexagon c3(c2);
std::cout << "Hexagon 3 coordinates are: " << std::endl;
c3.Print(std::cout);
std::cout << "Hexagon 3 area is:" << c3.Area() << std::endl;
}

```