

**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 Молчанов Владислав Дмитриевич, группа М80-208Б-20
Преподаватель Дорохов Евгений Павлович

Условие

Задание: Вариант 13: 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 стандарт-ного 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 10 файлах:

1. 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/pentagon.cpp: реализация класса пятиугольника, наследующегося от figures
9. include/hexagon.cpp: реализация класса шестиугольника, наследующегося от figures
10. include/rhombus.cpp: реализация класса ромба, наследующегося от figure

Во время выполнения лабораторной работы программа не нуждалась в отладке, все ошибки компиляции были исправлены с первой попытки. После их исправления программа работала так, как было задумано изначально.

Недочеты

Во время выполнения лабораторной работы недочетов в программе обнаружено не было.

Выводы:

Основная цель лабораторной работы №3 - знакомство с парадигмой объектно-ориентированного программирования на языке C++. Могу сказать, что справился с этой целью весьма успешно: усвоил “3 кита ООП”: полиморфизм, наследование, инкапсуляция, освоил базовые понятия ООП, такие как классы, методы, конструкторы, деструкторы... Ознакомился с ключевыми словами virtual, friend, private, public... Повторил тему “директивы условной компиляции”, “перегрузка функций/операторов”, работа со стандартными потоками ввода-вывода. **Лабораторная работа №3 прошла для меня успешно.**
Исходный код

figure.h

```
#ifndef FIGURE_H
#define FIGURE_H

#include "point.h"

class Figure {
public:
    virtual void Print() = 0;
```

```
virtual double Area() = 0;
virtual size_t VertexesNumber() = 0;
};
```

```
#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 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::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;  
}
```

pentagon.h

```
#ifndef PENTAGON_H
```

```
#define PENTAGON_H
```

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

```
#include <iostream>
```

```
using namespace std;
```

```
class Pentagon : public Figure {
```

```
public:
```

```
Pentagon(istream &is);
```

```
void Print();
```

```
double Area();
```

```
size_t VertexesNumber();
```

```
private:
```

```
Point a, b, c, d, e;
```

```
};
```

```
#endif
```

pentagon.cpp

```
#include "pentagon.h"
#include <cmath>
using namespace std;

Pentagon::Pentagon(istream &is){
    cout << "Enter all data: " << endl;
    cin >> a;
    cin >> b;
    cin >> c;
    cin >> d;
    cin >> e;
    cout << "Pentagon created via istream" << endl;
}

void Pentagon::Print(){
    cout << "Pentagon:"<< a << " " << b << " " << c << " " << d << " " << e <<endl;
}

double Pentagon::Area(){
    return 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;
}

size_t Pentagon::VertexesNumber(){
    size_t h = 5;
    return h;
}
```

rhombus.h

```
#include "rhombus.h"
```

```
using namespace std;
```

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

```
    cout << "Enter all data: " << endl;
```

```
cin >> a;
```

```
cin >> b;
```

```
cin >> c;
```

```
cin >> d;
```

```
cout << "Rhombus created via istream" << endl;
```

```
}
```

```
void Rhombus::Print() {
```

```
    cout << "Rhombus"<< a << " " << b << " " << c << " " << d << endl;
```

```
}
```

```
double Rhombus::Area() {
```

```
    return 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;
```



```
}

size_t Rhombus::VertexesNumber() {

    size_t h = 4;

    return h;

}
```

rhombus.cpp

```
#include "pentagon.h"

#include <cmath>

using namespace std;

Pentagon::Pentagon(istream &is){

    cout << "Enter all data: " << endl;

    cin >> a;
```

```
cin >> b;
```

```
cin >> c;
```

```
cin >> d;
```

```
cin >> e;
```

```
cout << "Pentagon created via istream" << endl;
```

```
}
```

```
void Pentagon::Print() {
```

```
    cout << "Pentagon:" << a << " " << b << " " << c << " " << d << " " << e << endl;
```

```
}
```

```
double Pentagon::Area() {
```

```
    return 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;
```

```
}
```

```
size_t Pentagon::VertexesNumber() {
```

```
    size_t h = 5;
```

```
    return h;
```

```
}
```

hexagon.h

```
#ifndef HEXAGON_H
```

```
#define HEXAGON_H
```

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

```
#include <iostream>
```

```
class Hexagon : public Figure {
```

```
public:
```

```
    Hexagon(std::istream &InputStream);
```

```
    virtual ~Hexagon();
```

```

size_t VertexesNumber();

double Area();

void Print(std::ostream &OutputStream);
private:

Point a;

Point b;

Point c;

Point d;

Point e;

Point f;

};

#endif

```

hexagon.cpp

```

#include "hexagon.h"

#include <cmath>

Hexagon::Hexagon(std::istream &InputStream)
{
    InputStream >> a;

    InputStream >> b;

    InputStream >> c;

    InputStream >> d;

    InputStream >> e;

    InputStream >> f;

    std::cout << "Hexagon that you wanted to create has been created"
<< std::endl;

}

void Hexagon::Print(std::ostream &OutputStream) {
    OutputStream << "Hexagon: ";

    OutputStream << a << " " << b << " " << c << " " << d << " " << e
<< " " << f << std::endl;
}

```

```

    }

    size_t Hexagon::VertexesNumber() {

        size_t number = 6;

        return number;

    }

    double Hexagon::Area() {

        double q = 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());

        double s = q / 2;

        return s;

    }

    Hexagon::~Hexagon() {

        std::cout << "My friend, your hexagon has been deleted" <<
std::endl;

    }

```

main.cpp

```

#include <iostream>
#include "rhombus.h"
#include "hexagon.h"
#include "pentagon.h"
using namespace std;
int main(){

    Rhombus Romb(cin);

    cout << "Area is: " << Romb.Area() << "\n";

    Romb.Print();

    Romb.VertexesNumber();

    cout << "\n";

    Rhombus Hexagon(cin);

    cout << "Area is: " << Hexagon.Area() << "\n";

```

```
Hexagon.Print();
Hexagon.VertexesNumber();
cout << "\n";
Pentagon Pentagon(cin);
cout << "Area is: " << Pentagon.Area() << "\n";
Pentagon.Print();
Pentagon.VertexesNumber();
cout << "\n";

return 0;
}
```