МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСТИТЕТ)

ЛАБОРАТОРНАЯ РАБОТА №3

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

Студент *Ханнанов Руслан Маратович, группа М8О-208Б-20*

Преподаватель *Дорохов Евгений Павлович*

Условие

Задание: Вариант 22: Пятиугольник, Шестиугольник, Восьмиугольник. 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 10 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/pentagon.h: описание класса пятиугольника, наследующегося от figures
5. include/hexagon.h: описание класса шестиугольника, наследующегося от figures
6. include/octagon.h: описание класса восьмиугольника, 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 прямоугольника, наследующегося от figures
10. include/octagon.cpp: реализация класса восьмиугольника, наследующегося от figures

Протокол работы

0 0 0 2 1 3 2 3 3 0
Pentagon created via istream
Pentagon:(0, 0) (0, 2) (1, 3) (2, 3) (3, 0)
Area is 7
Number of vertexes is 5
0 0 0 3 2 6 5 6 7 3 7 0
Hexagon created via istream
Hexagon:(0, 0) (0, 3) (2, 6) (5, 6) (7, 3) (7, 0)
Area is 36
Number of vertexes is 6
0 0 0 4 4 8 6 10 8 10 10 8 14 4 14 0
Octagon created via istream
Octagon:(0, 0) (0, 4) (4, 8) (6, 10) (8, 10) (10, 8) (14, 4) (14, 0)
Area is 104
Number of vertexes is 8
Object octagon Octagon:(0, 0) (0, 4) (4, 8) (6, 10) (8, 10) (10, 8) (14, 4) (14, 0)
deleted
Object hexagon Hexagon:(0, 0) (0, 3) (2, 6) (5, 6) (7, 3) (7, 0)
deleted
Object Pentagon Pentagon:(0, 0) (0, 2) (1, 3) (2, 3) (3, 0)
deleted

Дневник отладки

Недочёты

Недочетов не заметил

Выводы

В процессе выполнения данной лабораторной работы я познакомился с основами ООП и его принципами, такими как наследование, абстракция, инкапсуляция и полиморфизм. В работе также нужно было реализовать перегрузку оператора. В качестве результата я написал три класса фигур, наследуемых от одного начального класса и показал примеры их использования.

Исходный код:

figure.h

```
#ifndef OOP_LAB_1_FIGURE_H
#define OOP_LAB_1_FIGURE_H
#include "point.h"

class Figure {
private:
    virtual void Print(std::ostream& os) = 0;
    virtual double Area() = 0;
    virtual size_t VertexesNumber() = 0;
};

#endif
```

point.h

```
#ifndef OOP_LAB_1_POINT_H
#define OOP_LAB_1_POINT_H
#include <iostream>

class Point {
public:
    Point();
    Point(std::istream &is);
    Point(double x, double y);

    double dist(Point& other);
    friend double getx(Point& p);
    friend double gety(Point& p);

    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.cpp

```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);
}

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;
}

double getx(Point& p) {
    return p.x_;
}

double gety(Point& p) {
    return p.y_;
}
```

hexagon.h

```cpp
#ifndef OOP_LAB_1_HEXAGON_H
#define OOP_LAB_1_HEXAGON_H

#include "figure.h"
```

```
#include <iostream>

class Hexagon : public Figure {
public:
    Hexagon();
    Hexagon(Point v1,Point v2,Point v3,Point v4,Point v5,Point v6);
    explicit Hexagon(std::istream &is);
    Hexagon(Hexagon &other);

    void Print(std::ostream& os) override;
    double Area() override;
    size_t VertexesNumber() override;

    ~Hexagon();
private:
    Point v1,v2,v3,v4,v5,v6;
};
#endif
```

hexagon.cpp

```
#include "hexagon.h"

#include <cmath>

Hexagon::Hexagon(): v1(0,0),v2(0,0),v3(0,0),v4(0,0),v5(0,0),v6(0,0){
    std::cout << "Default hexagon created" << std::endl;
}

Hexagon::Hexagon(Point v_1,Point v_2, Point v_3, Point v_4, Point v_5, Point v_6):
        v1(v_1), v2(v_2), v3(v_3), v4(v_4), v5(v_5), v6(v_6)
{
    std::cout << "Hexagon created" << std::endl;
}

Hexagon::Hexagon(Hexagon& other):
        Hexagon(other.v1,other.v2,other.v3,other.v4,other.v5,other.v6)
{
    std::cout << "Made copy of hexagon";
}

Hexagon::Hexagon(std::istream &is) {
    is >> v1 >> v2 >> v3 >> v4 >> v5 >> v6;
```

```
    std::cout << "Hexagon created via istream" << std::endl;
}

void Hexagon::Print(std::ostream& os) {
    os << "Hexagon:" << v1 << v2 << v3 << v4 << v5 << v6 << "\n";
}

Hexagon::~Hexagon() {
    std::cout << "Object hexagon ";
    Print(std::cout);
    std::cout << "deleted" << std::endl;
}

double Hexagon::Area() {
    Point ar[6];
    ar[0] = v1;
    ar[1] = v2;
    ar[2] = v3;
    ar[3] = v4;
    ar[4] = v5;
    ar[5] = v6;
    double res = 0;
    for (unsigned i = 0; i < 6; i++) {
        Point p = i ? ar[i-1] : ar[5];
        Point q = ar[i];
        res += (getx(p) - getx(q)) * (gety(p) + gety(q));
    }
    return fabs(res) / 2;
}

size_t Hexagon::VertexesNumber() {
    return 6;
}
```

octagon.h

```
#ifndef OOP_LAB_1_OCTAGON_H
#define OOP_LAB_1_OCTAGON_H
#include "figure.h"

#include <iostream>

class Octagon : public Figure {
```

```
public:
    Octagon();
    Octagon(Point v1,Point v2,Point v3,Point v4,Point v5,Point v6,Point v7, Point v8);
    explicit Octagon(std::istream &is);
    Octagon(Octagon &other);

    void Print(std::ostream& os) override;
    double Area() override;
    size_t VertexesNumber() override;

    ~Octagon();
private:
    Point v1,v2,v3,v4,v5,v6,v7,v8;
};
#endif
```

octagon.cpp

```
#include <cmath>

#include "octagon.h"

Octagon::Octagon(): v1(0,0),v2(0,0),v3(0,0),v4(0,0),v5(0,0),v6(0,0),v7(0,0),v8(0,0){
    std::cout << "Default octagon created" << std::endl;
}

Octagon::Octagon(Point v_1,Point v_2, Point v_3, Point v_4, Point v_5, Point v_6, Point
        v1(v_1), v2(v_2), v3(v_3), v4(v_4), v5(v_5), v6(v_6), v7(v_7), v8(v_8)
{
    std::cout << "Octagon created" << std::endl;
}

Octagon::Octagon(Octagon& other):
        Octagon(other.v1,other.v2,other.v3,other.v4,other.v5,other.v6,other.v7,other.v8)
{
    std::cout << "Made copy of octagon";
}

Octagon::Octagon(std::istream &is) {
    is >> v1 >> v2 >> v3 >> v4 >> v5 >> v6 >> v7 >> v8;
    std::cout << "Octagon created via istream" << std::endl;
}
```

```cpp
void Octagon::Print(std::ostream& os) {
    os << "Octagon:" << v1 << v2 << v3 << v4 << v5 << v6 << v7 << v8 << "\n";
}

Octagon::~Octagon() {
    std::cout << "Object octagon ";
    Print(std::cout);
    std::cout << "deleted" << std::endl;
}

double Octagon::Area() {
    Point ar[8];
    ar[0] = v1;
    ar[1] = v2;
    ar[2] = v3;
    ar[3] = v4;
    ar[4] = v5;
    ar[5] = v6;
    ar[6] = v7;
    ar[7] = v8;
    double res = 0;
    for (unsigned i = 0; i < 8; i++) {
        Point p = i ? ar[i-1] : ar[7];
        Point q = ar[i];
        res += (getx(p) - getx(q)) * (gety(p) + gety(q));
    }
    return fabs(res) / 2;
}

size_t Octagon::VertexesNumber() {
    return 8;
}
```

pentagon.h

```cpp
#ifndef OOP_LAB_1_PENTAGON_H
#define OOP_LAB_1_PENTAGON_H

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

class Pentagon : public Figure {
```

```
public:
    Pentagon();
    Pentagon(Point v1,Point v2,Point v3,Point v4,Point v5);
    explicit Pentagon(std::istream &is);
    Pentagon(Pentagon &other);

    void Print(std::ostream& os) override;
    double Area() override;
    size_t VertexesNumber() override;

    ~Pentagon();
private:
    Point v1,v2,v3,v4,v5;
};
#endif
```

pentagon.cpp

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

Pentagon::Pentagon(): v1(0,0),v2(0,0),v3(0,0),v4(0,0),v5(0,0){
    std::cout << "Default pentagon created" << std::endl;
}

Pentagon::Pentagon(Point v_1,Point v_2, Point v_3, Point v_4, Point v_5):
    v1(v_1), v2(v_2), v3(v_3), v4(v_4), v5(v_5)
{
    std::cout << "Pentagon created" << std::endl;
}

Pentagon::Pentagon(Pentagon& other):
Pentagon(other.v1,other.v2,other.v3,other.v4,other.v5)
{
    std::cout << "Made copy of pentagon";
}

Pentagon::Pentagon(std::istream &is) {
    is >> v1 >> v2 >> v3 >> v4 >> v5;
    std::cout << "Pentagon created via istream" << std::endl;
}

void Pentagon::Print(std::ostream& os) {
```

```
    os << "Pentagon:" << v1 << v2 << v3 << v4 << v5 << "\n";
}

Pentagon::~Pentagon() {
    std::cout << "Object Pentagon ";
    Print(std::cout);
    std::cout << "deleted" << std::endl;
}

double Pentagon::Area() {
    Point ar[5];
    ar[0] = v1;
    ar[1] = v2;
    ar[2] = v3;
    ar[3] = v4;
    ar[4] = v5;
    double res = 0;
    for (unsigned i = 0; i < 5; i++) {
        Point p = i ? ar[i-1] : ar[4];
        Point q = ar[i];
        res += (getx(p) - getx(q)) * (gety(p) + gety(q));
    }
    return fabs(res) / 2;
}

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

main.cpp

```
#include <iostream>
#include "pentagon.h"
#include "hexagon.h"
#include "octagon.h"
int main() {

    //Pentagon:
    Pentagon p(std::cin);
    p.Print(std::cout);
    std::cout << "Area is " << p.Area() << "\n";
    std::cout << "Number of vertexes is " << p.VertexesNumber() << "\n";
```

```
    //Hexagon:
    Hexagon h(std::cin);
    h.Print(std::cout);
    std::cout << "Area is " << h.Area() << "\n";
    std::cout << "Number of vertexes is " << h.VertexesNumber() << "\n";

    //Octagon:
    Octagon o(std::cin);
    o.Print(std::cout);
    std::cout << "Area is " << o.Area() << "\n";
    std::cout << "Number of vertexes is " << o.VertexesNumber() << "\n";

    return 0;
}
/* Test's:
Pentagon (Area = 7):
0 0 0 2 1 3 2 3 3 0
Hexagon (Area = 36):
0 0 0 3 2 6 5 6 7 3 7 0
Octagon (Area = 104):
0 0 0 4 4 8 6 10 8 10 10 8 14 4 14 0
*/
```