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 *Белоусов Егор Владимирович, группа М8О-207Б-20*

Преподаватель *Дорохов Евгений Павлович*

Условие

Задание:

Вариант 7: 6-угольник, 8-угольник, Треугольник. 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 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(в случае, если многоугольник не является выпуклым возвращает -1);
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. figure.h: описание абстрактного класса фигур
3. point.h: описание класса точки
4. triangle.h: описание класса треугольника, наследующегося от figure
5. hexagon.h: описание класса 6-угольника, наследующегося от figure

6. octagon.h: описание класса 8-угольника, наследующегося от figure
7. point.cpp: реализация класса точки
8. triangle.cpp: реализация класса треугольника, наследующегося от figure
9. hexagon.cpp: реализация класса 6-угольника, наследующегося от figure
10. octagon.cpp: реализация класса 8-угольника, наследующегося от figure

Дневник отладки

D:\oop\lab1\cmake-build-debug\lab1.exe

Enter a triangle:

-1.0 0.0

0.0 1.0

1.0 0.0

Vertexes of triangle: 3

Triangle: (-1, 0) (0, 1) (1, 0)

Area of triangle: 1

Enter a hexagon:

0.0 2.0

2.0 4.0

4.0 4.0

6.0 2.0

4.0 0.0

2.0 0.0

Vertexes of hexagon: 6

Hexagon: (0, 2) (2, 4) (4, 4) (6, 2) (4, 0) (2, 0)

Area of hexagon: 16

Enter a octagon:

0.0 2.0

0.0 4.0

2.0 6.0

4.0 6.0

6.0 4.0

6.0 2.0

4.0 0.0

2.0 0.0

Vertexes of octagon: 8

Octagon: (0, 2) (0, 4) (2, 6) (4, 6) (6, 4) (6, 2) (4, 0) (2, 0)

Area of octagon: 28

Process finished with exit code 0

D:\oop\lab1\cmake-build-debug\lab1.exe

Enter a triangle:

-3.5 4.7

2.5 1.2

0.0 -10.0

Vertexes of triangle: 3

Triangle: (-3.5, 4.7) (2.5, 1.2) (0, -10)

Area of triangle: 37.975

Enter a hexagon:

1.0 0.0

0.0 1.0

1.0 2.0

0.5 1.0

1.0 1.0

1.5 0.5

Vertexes of hexagon: 6

Hexagon: (1, 0) (0, 1) (1, 2) (0.5, 1) (1, 1) (1.5, 0.5)

Area of hexagon: -1

Enter a octagon:

0.0 2.0

6.0 2.0

4.0 1.0

5.0 0.0

4.0 0.5

3.0 0.0

2.0 1.0

1.0 0.0

Vertexes of octagon: 8

Octagon: (0, 2) (6, 2) (4, 1) (5, 0) (4, 0.5) (3, 0) (2, 1) (1, 0)

Area of octagon: -1

Process finished with exit code 0

Недочёты

По моему мнению недочетов нет.

Выводы

В данной лабораторной работе я познакомился с простейшими понятиями объектно-ориентированного программирования: я реализовал несколько классов, использовал при создании наследование. Также я понял, что объектно-ориентированный подход помогает писать более структурированный и понятный код, что очень полезно в промышленных условиях.

Исходный код

figure.h

```
#ifndef LAB1_FIGURE_H
#define LAB1_FIGURE_H

#include <iostream>

class Figure {
public:
    virtual double Area() = 0;

    virtual size_t VertexesNumber() = 0;

    virtual void Print(std::ostream &os) = 0;
};


#endif //LAB1_FIGURE_H
```

point.h

```
#ifndef LAB1_POINT_H
#define LAB1_POINT_H

#include <iostream>

class Point {
private:
    double _x, _y;
public:
    Point();

    Point(double x, double y);

    friend double operator*(Point &a, Point &b);

    friend Point operator-(Point &a, Point &b);

    friend double dist(Point &a, Point &b);

    friend std::istream &operator>>(std::istream &is, Point &p);

    friend std::ostream &operator<<(std::ostream &os, Point &p);
};


#endif //LAB1_POINT_H
```

point.cpp

```
#include "point.h"

Point::Point() : _x(0.0), _y(0.0) {}

Point::Point(double x, double y) : _x(x), _y(y) {}
```

```cpp
double operator*(Point &a, Point &b) {
    return a._x * b._y - b._x * a._y;
}

Point operator-(Point &a, Point &b) {
    Point c;
    c._x = b._x - a._x;
    c._y = b._y - a._y;
    return c;
}

double dist(Point &a, Point &b) {
    double dx = (b._x - a._x);
    double mid = (b._y + a._y) / 2.0;
    return dx * mid;
}

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;
}
```

triangle.h

```cpp
#ifndef LAB1_TRIANGLE_H
#define LAB1_TRIANGLE_H

#include "figure.h"
#include "point.h"

#include <iostream>

class Triangle : public Figure {
private:
    static const int size = 3;
    Point P[size];
public:
    Triangle();

    Triangle(std::istream &is);

    double Area() override;

    size_t VertexesNumber() override;

    void Print(std::ostream &os) override;

};


#endif //LAB1_TRIANGLE_H
```

triangle.cpp

```cpp
#include "triangle.h"

Triangle::Triangle() {}

Triangle::Triangle(std::istream &is) {
    for (size_t i = 0; i < size; ++i) {
        is >> P[i];
    }
}

double Triangle::Area() {
    double area = 0.0;
    for (size_t i = 0; i < size; ++i) {
        area += dist(P[i], P[(i + 1) % size]);
    }
    if (area < 0)area = -1.0 * area;
    return area;
}

size_t Triangle::VertexesNumber() {
    return size;
}

void Triangle::Print(std::ostream &os) {
    os << "Triangle: ";
    for (size_t i = 0; i < size; ++i) {
        os << P[i] << " ";
    }
    os << "\n";
}
```

hexagon.h

```cpp
#ifndef LAB1_HEXAGON_H
#define LAB1_HEXAGON_H

#include "figure.h"
#include "point.h"

class Hexagon : public Figure {
private:
    static const size_t size = 6;
    Point P[size];
public:
    Hexagon();

    Hexagon(std::istream &is);

    double Area() override;

    size_t VertexesNumber() override;

    void Print(std::ostream &os) override;
};


#endif //LAB1_HEXAGON_H
```

hexagon.cpp

```cpp
#include "hexagon.h"

Hexagon::Hexagon() {}

Hexagon::Hexagon(std::istream &is) {
    for (size_t i = 0; i < size; ++i) {
        is >> P[i];
    }
}

double Hexagon::Area() {
    const double eps = 1e-9;
    bool convex = true;
    for (size_t i = 0; i < size - 1; ++i) {
        Point a = P[(i + 1) % size] - P[i];
        Point b = P[(i + 2) % size] - P[(i + 1) % size];
        if (a * b > eps) {
            convex = false;
        }
    }
    if (!convex) {
        return -1.0;
    }
    double area = 0.0;
    for (size_t i = 0; i < size; ++i) {
        area += dist(P[i], P[(i + 1) % size]);
    }
    if (area < 0.0)area = -1.0 * area;
    return area;
}

size_t Hexagon::VertexesNumber() {
    return size;
}

void Hexagon::Print(std::ostream &os) {
    os << "Hexagon: ";
    for (size_t i = 0; i < size; ++i) {
        os << P[i] << " ";
    }
    os << "\n";
}
```

octagon.h

```cpp
#ifndef LAB1_OCTAGON_H
#define LAB1_OCTAGON_H

#include "figure.h"
#include "point.h"


class Octagon : public Figure {
private:
    static const size_t size = 8;
```

```
    Point P[size];
public:
    Octagon();

    Octagon(std::istream &is);

    double Area() override;

    size_t VertexesNumber() override;

    void Print(std::ostream &os) override;
};


#endif //LAB1_OCTAGON_H
```

octagon.cpp

```
#include "octagon.h"

Octagon::Octagon() {}

Octagon::Octagon(std::istream &is) {
    for (size_t i = 0; i < size; ++i) {
        is >> P[i];
    }
}

double Octagon::Area() {
    const double eps = 1e-9;
    bool convex = true;
    for (size_t i = 0; i < size - 1; ++i) {
        Point a = P[(i + 1) % size] - P[i];
        Point b = P[(i + 2) % size] - P[(i + 1) % size];
        if (a * b > eps) {
            convex = false;
        }
    }
    if (!convex) {
        return -1.0;
    }
    double area = 0.0;
    for (size_t i = 0; i < size; ++i) {
        area += dist(P[i], P[(i + 1) % size]);
    }
    if (area < -eps)area = -1.0 * area;
    return area;
}

size_t Octagon::VertexesNumber() {
    return size;
}

void Octagon::Print(std::ostream &os) {
    os << "Octagon: ";
    for (size_t i = 0; i < size; ++i) {
        os << P[i] << " ";
    }
    os << "\n";
}
```

main.cpp

```cpp
#include "triangle.h"
#include "hexagon.h"
#include "octagon.h"

int main() {
    std::cout << "Enter a triangle: \n";
    Triangle T1(std::cin);
    std::cout << "Vertexes of triangle: " << T1.VertexesNumber() << "\n";
    T1.Print(std::cout);
    std::cout << "Area of triangle: " << T1.Area() << "\n";
    std::cout << "Enter a hexagon: \n";
    Hexagon H1(std::cin);
    std::cout << "Vertexes of hexagon: " << H1.VertexesNumber() << "\n";
    H1.Print(std::cout);
    std::cout << "Area of hexagon: " << H1.Area() << "\n";
    std::cout << "Enter a octagon: \n";
    Octagon O1(std::cin);
    std::cout << "Vertexes of octagon: " << O1.VertexesNumber() << "\n";
    O1.Print(std::cout);
    std::cout << "Area of octagon: " << O1.Area() << "\n";
    return 0;
}
```