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: Москвин Артём Артурович, группа М80-208Б-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 №15:

Фигура 1: Шестиугольник (Hexagon)

Фигура 2: Восьмиугольник (Octagon)

Фигура 3: Треугольник (Triangle)

Описание программы

Исходный код разделён на 10 файлов:

- main.cpp: основная программа, взаимодействие с пользователем посредством команд из меню
- include/figure.h: описание абстрактного класса фигур
- include/point.h: описание класса точки
- include/hexagon.h: описание класса шестиугольника, наследующегося от figures
- include/octagon.h: описание класса восьмиугольника, наследующегося от figures
- include/triangle.h: описание класса треугольника, наследующегося от figures
- include/point.cpp: реализация класса точки
- include/hexagon.cpp: реализация класса шестиугольника, наследующегося от figures
- include/octagon.cpp: реализация класса восьмиугольника, наследующегося от figures
- include/triangle.cpp: реализация класса треугольника, наследующегося от figure

Дневник отладки:

Программа в отладке не нуждалась, необходимый функционал был реализован довольно быстро и безошибочно.

Вывод:

Данная лабораторная работа познакомила меня с оставшимися двумя из трех китов ООП: если с инкапсуляцией я уже знаком, то благодаря ЛР №3 я знаю, что такое полиморфизм и наследование. Достичь этого получилось при помощи реализации класса “Figure”. Дело в том, что от этого класса далее наследуются наши пятиугольники, шестиугольники и восьмиугольники. А полиморфизм достигается за счет виртуальных функций (ключевое слово **virtual**). Описав виртуальные методы **Print, Area, VertexesNumber,** мы автоматически позволили сами же себе реализовать эти методы в каждом классе многоугольников по-разному. В этом и заключается принцип полиморфизма в данной ЛР.

Исходный код

figure.h

```
#ifndef FIGURE_H
#define FIGURE_H

#include <iostream>


class Figure {
public:
    virtual void Print(std::ostream& os) = 0;
    virtual double Area() = 0;
    virtual size_t VertexesNumber() = 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 dist(Point& other);
```

```
  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::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;
}
```

octagon.h

```
#ifndef OCTAGON_H
#define OCTAGON_H

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

#include <iostream>
```

```
class Octagon : public Figure {
public:
    Octagon();
    Octagon(Point a, Point b, Point c, Point d, Point e, Point f, Point g, Point h);
    Octagon(std::istream &is);
    Octagon(const Octagon& other);

    virtual ~Octagon();

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

private:
    Point A, B, C, D, E, F, G, H;
    size_t PointCount = 6;
    std::string ClassName = "Octagon";
};

#endif //f OCTAGON_H
```

octagon.cpp

```
#include "octagon.h"
#include "point.h"
#include <string>
#include "triangle.h"

#include <cmath>

Octagon::Octagon()
    : 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),
      G(0.0, 0.0),
      H(0.0, 0.0) {
  std::cout << "Default octagon created" << std::endl;
}

Octagon::Octagon(Point a, Point b, Point c, Point d, Point e, Point f, Point g, Point
h)
    : A(a),
      B(b),
```

```
      C(c),
      D(d),
      E(e),
      F(f),
      G(g),
      H(h) {
  std::cout << "Octagon created" << std::endl;
}

Octagon::Octagon(std::istream &is) {
  std::cout << "Enter data:" <<  std::endl;
  is >> A;
  is >> B;
  is >> C;
  is >> D;
  is >> E;
  is >> F;
  is >> G;
  is >> H;
  std::cout << "Octagon created via istream" << std::endl;
}

Octagon::Octagon(const Octagon& other)
    : Octagon(other.A, other.B, other.C, other.D, other.E, other.F, other.G, other.H) {
  std::cout << "Made copy of octagon" << std::endl;

}

void Octagon::Print(std::ostream& os) {
    os << ClassName << A << ", " << B << ", " << C << ", " << D << ", " << E << ", " <<
F << ", " << G << ", " << H << std::endl;
}

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

double Octagon::Area() {
    return Triangle(A, B, C).Area() + Triangle(A, C, D).Area() + Triangle(A, D,
E).Area() + Triangle(A, E, F).Area() + Triangle(A, F, G).Area() + Triangle(A, G,
H).Area();
}

Octagon::~Octagon() {
    std::cout << "Octagon deleted" << std::endl;
}
```

hexagon.h

```
#ifndef HEXAGON_H
#define HEXAGON_H

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

#include <iostream>


class Hexagon : public Figure {
public:
    Hexagon();
    Hexagon(Point a, Point b, Point c, Point d, Point e, Point f);
    Hexagon(std::istream &is);
    Hexagon(const Hexagon& other);

    virtual ~Hexagon();

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

private:
    Point A, B, C, D, E, F;
    size_t PointCount = 6;
    std::string ClassName = "Hexagon";
};

#endif //f HEXAGON_H
```

hexagon.cpp

```
#include "hexagon.h"
#include "point.h"
#include <string>
#include "triangle.h"

#include <cmath>

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(Point a, Point b, Point c, Point d, Point e, Point f)
    : A(a),
      B(b),
      C(c),
      D(d),
      E(e),
      F(f) {
  std::cout << "Hexagon created" << std::endl;
}

Hexagon::Hexagon(std::istream &is) {
  std::cout << "Enter data:" <<  std::endl;
  is >> A;
  is >> B;
  is >> C;
  is >> D;
  is >> E;
  is >> F;
  std::cout << "Hexagon created via istream" << std::endl;
}

Hexagon::Hexagon(const Hexagon& other)
    : Hexagon(other.A, other.B, other.C, other.D, other.E, other.F) {
  std::cout << "Made copy of hexagon" << std::endl;

}

void Hexagon::Print(std::ostream& os) {
    os << ClassName << A << ", " << B << ", " << C << ", " << D << ", " << E << ", " <<
F << std::endl;
}

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

double Hexagon::Area() {
    return Triangle(A, B, C).Area() + Triangle(A, C, D).Area() + Triangle(A, D,
F).Area() + Triangle(D, F, E).Area();
}

Hexagon::~Hexagon() {
    std::cout << "Hexagon deleted" << std::endl;
}
```

triangle.h

```
#ifndef TRIANGLE_H
```

```
#define TRIANGLE_H

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

#include <iostream>


class Triangle : public Figure {
public:
    Triangle();
    Triangle(Point a, Point b, Point c);
    Triangle(std::istream &is);
    Triangle(const Triangle& other);

    virtual ~Triangle();

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

private:
    Point A, B, C;
    size_t PointCount = 3;
    std::string ClassName = "Triangle";
};

#endif // TRIANGLE_H
```

triangle.cpp

```
#include "triangle.h"
#include "point.h"
#include <string>

#include <cmath>

Triangle::Triangle()
    : A(0.0, 0.0),
      B(0.0, 0.0),
      C(0.0, 0.0) {
  std::cout << "Default triangle created" << std::endl;
}
```

```
Triangle::Triangle(Point a, Point b, Point c)
    : A(a),
      B(b),
      C(c) {
  std::cout << "Triangle created" << std::endl;
}

Triangle::Triangle(std::istream &is) {
  std::cout << "Enter data:" <<  std::endl;
  is >> A;
  is >> B;
  is >> C;
  std::cout << "Triangle created via istream" << std::endl;
}

Triangle::Triangle(const Triangle& other)
    : Triangle(other.A, other.B, other.C) {
  std::cout << "Made copy of triangle" << std::endl;

}

void Triangle::Print(std::ostream& os) {
    os << ClassName << A << ", " << B << ", " << C << std::endl;
}

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

double Triangle::Area() {
    double p = (A.dist(B) + B.dist(C) + C.dist(A)) / 2.0;
    return std::sqrt(p * (p - A.dist(B)) * (p - B.dist(C)) * (p - C.dist(A)));
}

Triangle::~Triangle() {
    std::cout << "Triangle deleted" << std::endl;
}
```

main.cpp

```
#include <iostream>
#include "triangle.h"
#include "hexagon.h"
#include "octagon.h"

using namespace std;

int main()
{
```

```
    //Triangle

    Triangle TriangleA;
    TriangleA.Print(cout);
    std::cout << TriangleA.Area() << std::endl;
    Triangle TriangleB(Point(0.0, 0.0), Point(1.0, 1.0), Point(0.0, 1.0));
    TriangleB.Print(cout);
    std::cout << TriangleB.Area() << std::endl;
    Triangle TriangleC(cin);
    TriangleC.Print(cout);
    std::cout << TriangleC.Area() << std::endl;
    Triangle TriangleD(TriangleB);
    TriangleD.Print(cout);
    std::cout << TriangleD.Area() << std::endl;

    //Hexagon

    Hexagon HexagonA;
    HexagonA.Print(cout);
    std::cout << HexagonA.Area() << std::endl;
    Hexagon HexagonB(Point(0.0, 1.0), Point(1.0, 0.0), Point(2.0, 1.0), Point(2.0,
2.0), Point(1.0, 3.0), Point(0.0, 2.0));
    HexagonB.Print(cout);
    std::cout << HexagonB.Area() << std::endl;
    Hexagon HexagonC(cin);
    HexagonC.Print(cout);
    std::cout << HexagonC.Area() << std::endl;
    Hexagon HexagonD(HexagonB);
    HexagonD.Print(cout);
    std::cout << HexagonD.Area() << std::endl;

    //Octagon

    Octagon OctagonA;
    OctagonA.Print(cout);
    std::cout << OctagonA.Area() << std::endl;
    Octagon OctagonB(Point(0.0, 1.0), Point(0.0, 2.0), Point(3.0, 1.0), Point(3.0,
2.0), Point(2.0, 3.0), Point(1.0, 3.0), Point(0.0, 2.0), Point(0.0, 1.0));
    OctagonB.Print(cout);
    std::cout << OctagonB.Area() << std::endl;
    Octagon OctagonC(cin);
    OctagonC.Print(cout);
    std::cout << OctagonC.Area() << std::endl;
    Octagon OctagonD(OctagonB);
    OctagonD.Print(cout);
    std::cout << OctagonD.Area() << std::endl;

    return 0;
}
```

Пример работы:

```cpp
#include <iostream>
#include "triangle.h"
#include "hexagon.h"
#include "octagon.h"

using namespace std;

int main()
{

    //Triangle

    Triangle TriangleA;
    TriangleA.Print(cout);
    std::cout << TriangleA.Area() << std::endl;
    Triangle TriangleB(Point(0.0, 0.0), Point(1.0, 1.0), Point(0.0, 1.0));
    TriangleB.Print(cout);
    std::cout << TriangleB.Area() << std::endl;
    Triangle TriangleC(cin);
    TriangleC.Print(cout);
    std::cout << TriangleC.Area() << std::endl;
    Triangle TriangleD(TriangleB);
    TriangleD.Print(cout);
    std::cout << TriangleD.Area() << std::endl;

    //Hexagon

    Hexagon HexagonA;
    HexagonA.Print(cout);
    std::cout << HexagonA.Area() << std::endl;
    Hexagon HexagonB(Point(0.0, 1.0), Point(1.0, 0.0), Point(2.0, 1.0), Point(2.0, 2.0), Point(1.0, 3.0), Point(0.0, 2.0));
    HexagonB.Print(cout);
    std::cout << HexagonB.Area() << std::endl;
    Hexagon HexagonC(cin);
    HexagonC.Print(cout);
    std::cout << HexagonC.Area() << std::endl;
    Hexagon HexagonD(HexagonB);
    HexagonD.Print(cout);
    std::cout << HexagonD.Area() << std::endl;
```

```cpp
    //Octagon

    Octagon OctagonA;
    OctagonA.Print(cout);
    std::cout << OctagonA.Area() << std::endl;
    Octagon OctagonB(Point(0.0, 1.0), Point(0.0, 2.0), Point(3.0, 1.0), Point(3.0, 2.0), Point(2.0, 3.0), Point(1.0, 3.0), Point(0.0, 2.0), Point(0.0,
    OctagonB.Print(cout);
    std::cout << OctagonB.Area() << std::endl;
    Octagon OctagonC(cin);
    OctagonC.Print(cout);
    std::cout << OctagonC.Area() << std::endl;
    Octagon OctagonD(OctagonB);
    OctagonD.Print(cout);
    std::cout << OctagonD.Area() << std::endl;

    return 0;
}
```

```
Default triangle created
Triangle(0, 0), (0, 0), (0, 0)
0
Triangle created
Triangle(0, 0), (1, 1), (0, 1)
0.5
Enter data:
1 2 3 4 5 6
Triangle created via istream
Triangle(1, 2), (3, 4), (5, 6)
0
Triangle created
Made copy of triangle
Triangle(0, 0), (1, 1), (0, 1)
0.5
Default hexagon created
Hexagon(0, 0), (0, 0), (0, 0), (0, 0), (0, 0), (0, 0)
Triangle created
Triangle created
Triangle created
Triangle created
Triangle deleted
Triangle deleted
Triangle deleted
Triangle deleted
0
Hexagon created
Hexagon(0, 1), (1, 0), (2, 1), (2, 2), (1, 3), (0, 2)
Triangle created
Triangle created
Triangle created
Triangle created
Triangle deleted
Triangle deleted
Triangle deleted
Triangle deleted
4
Enter data:
1 2 3 4 5 6
7 8 9 10 11 12
Hexagon created via istream
Hexagon(1, 2), (3, 4), (5, 6), (7, 8), (9, 10), (11, 12)
```

```
Default octagon created
Octagon(0, 0), (0, 0), (0, 0), (0, 0), (0, 0), (0, 0), (0, 0), (0, 0)
Triangle created
Triangle created
Triangle created
Triangle created
Triangle created
Triangle created
Triangle deleted
Triangle deleted
Triangle deleted
Triangle deleted
Triangle deleted
Triangle deleted
0
Octagon created
Octagon(0, 1), (0, 2), (3, 1), (3, 2), (2, 3), (1, 3), (0, 2), (0, 1)
Triangle created
Triangle created
Triangle created
Triangle created
Triangle created
Triangle created
Triangle deleted
Triangle deleted
Triangle deleted
Triangle deleted
Triangle deleted
Triangle deleted
6.5
Enter data:
1 2 3 4 5 6
7 8 9 10 11 12
13 14 15 16 17 18
Octagon created via istream
Octagon(1, 2), (3, 4), (5, 6), (7, 8), (9, 10), (11, 12), (13, 14), (15, 16)
```