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 *Каширин Кирилл Дмитриевич, группа М8О-208Б-20*

Преподаватель *Дорохов Евгений Павлович*

Условие

Задание: Вариант 7: Шестиугольник, Восьмиугольник, 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 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 11 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/triangle.h: описание класса треугольника, 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/triangle.cpp: реализация класса треугольника, наследующегося от figures
9. include/hexagon.cpp: реализация класса прямоугольника, наследующегося от figures
10. include/octagon.cpp: реализация класса восьмиугольника, наследующегося от figures

Дневник отладки

Недочёты

Недочетов не заметил

Выводы

Данная лабораторная работа помогла мне разобраться с основами ООП, принципами ООП, такими как наследование, инкапсуляция, полиморфизм, абстракция, а также с понятием пререзгрузка оператора. Я реализовал несколько классов данных фигуры и функции, которые требовались в задании.

Исходный код:

figure.h

```
#ifndef FIGURE_H
#define FIGURE_H
#include "point.h"
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);

  double x();
  double y();

private:
  double x_;
  double y_;
};

#endif // POINT_H
```

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 Point::x(){
  return x_;
}

double Point::y(){
  return y_;
}
```

hexagon.h

```cpp
#ifndef HEXAGON_H
#define HEXAGON_H
#include "figure.h"
#include <iostream>
```

```
class Hexagon : public Figure {
    public:
    Hexagon(std::istream &is);

    virtual ~Hexagon();

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

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

#endif // HEXAGON_H
```

hexagon.cpp

```
#include "hexagon.h"
#include <cmath>
#include "point.h"

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

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

double Hexagon::Area() {
    return 0.5*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
}
size_t Hexagon::VertexesNumber(){
    return 6;
}
Hexagon::~Hexagon() {
    std::cout << "Hexagon deleted" << std::endl;
}
```

octagon.h

```cpp
#ifndef OCTAGON_H
#define OCTAGON_H
#include "figure.h"
#include <iostream>

class Octagon : public Figure {
    public:
    Octagon(std::istream &is);

    virtual ~Octagon();

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

    private:
    Point a, b, c, d, e, f, g, h;
};

#endif // OCTAGON_H
```

octagon.cpp

```cpp
#include "octagon.h"
#include <cmath>

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

void Octagon::Print(std::ostream &os) {
    os << "Octagon:" << a << b << c << d << e << f << g << h << std::endl;
}

double Octagon::Area() {
    return 0.5*abs(a.x() * b.y() + b.x() * c.y() + c.x() * d.y() + d.x() * e.y() + e.x()
}

size_t Octagon::VertexesNumber(){
    return 8;
}
Octagon::~Octagon() {
    std::cout << "Octagon deleted" << std::endl;
}
```

triagnle.h

```
#ifndef TRIANGLE_H
#define TRIANGLE_H
#include "figure.h"
#include <iostream>

class Triangle : public Figure {
    public:
    Triangle(std::istream &is);

    virtual ~Triangle();

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

    private:
    Point a, b, c, d, e, f, g, h;
};

#endif // Triangle_H
```

triangle.cpp

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

Triangle::Triangle(std::istream &is) {
    is >> a >> b >> c;
}

void Triangle::Print(std::ostream &os) {
    os << "Triangle:" << a << b << c << std::endl;
}

double Triangle::Area() {
    return 0.5*abs((b.x()-a.x())*(c.y()-a.y())-(c.x()-a.x()*(b.y()-a.y())));
}
size_t Triangle::VertexesNumber(){
    return 3;
}
Triangle::~Triangle() {
```

```
    std::cout << "Triangle deleted" << std::endl;
}
```

main.cpp

```
#include <iostream>
#include "hexagon.h"
#include "octagon.h"
#include "triangle.h"
using namespace std;

int main() {
    Hexagon a(std::cin);
    std::cout << "Square = " << a.Area() << std::endl;
    a.Print(std::cout);

    Octagon b(std::cin);
    std::cout << "Square = " << a.Area() << std::endl;
    b.Print(std::cout);

    Triangle c(std::cin);
    std::cout << "Square = " << c.Area() << std::endl;
    c.Print(std::cout);
}
```