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 *Соколов Даниил Витальевич, группа М8О-207Б-20*
Преподаватель *Дорохов Евгений Павлович*

Условие

Задание: Вариант 23: Треугольник, шестиугольник, Восьмиугольник. Необходимо спроекти- ровать 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 раздель- ных 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 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/octagon.h: описание класса восьмиугольника, наследующегося от figures
6. include/hexagon.h: описание класса шестиугольника
7. include/point.cpp: реализация класса точки
8. include/triangle.cpp: реализация класса треугольника, наследующегося от figures
9. include/octagon.cpp: реализация класса восьмиугольника, наследующегося от figures

10. include/hexagon.cpp: реализация класса шестиугольника, наследующегося от rectangle

Дневник отладки

Не было никаких ошибок

Недочёты

Считаю, что программу можно улучшить, добавив интерактивное меню для тестирова- ния всех функций программы.

Выводы

В ходе лабораторной работы удалось поработать с парадигмой объектно-ориентированного программирования - наследование, полиморфизм, инкапсуляция и абстакция. В современном мире больших проектов, enterprise-разработки, ООП является единственной приемлемой парадигмой разработки, безукоризненно выигры вающей конкуренцию у процедурного программирования.

Исходный код

figure.h

```
#ifndef FIGURE_H
#define FIGURE_H

#include "point.h"

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

  friend std::istream& operator>>(std::istream& ins, Point& p);
  friend std::ostream& operator<<(std::ostream& ssd, Point& p);

private:
  double x_;
  double y_;
};

#endif
```

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);
}

double Point::getX()
{
   return x_;
}

double Point::getY()
{
   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;
```

}

main.cpp

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

int main()
{

   Hexagon a(std::cin);
   Octagon b(std::cin);
   Triangle c(std::cin);

   a.Print(std::cout);
   b.Print(std::cout);
   c.Print(std::cout);

   return 0;
}
```

hexagon.h

```
#ifndef HEXAGON_H
#define HEXAGON_H

#include "figure.h"

class Hexagon : Figure
{
public:
   Hexagon(std::istream& ins);
   size_t VertexesNumber();
   double Area();
   void Print(std::ostream& ssd);

private:
   Point a_, b_, c_;
   Point d_, e_, f_,;
};

#endif
```

hexagon.cpp

```
#include "hexagon.h"

Hexagon::Hexagon(std::istream& ins)
{
    std::cin >> a_ >> b_ >> c_ >> d_;
    std::cin >> e_ >> f_;
}

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

double Hexagon::Area()
{
    return 0.5 * abs((a_.getX() * b_.getY() + b_.getX() * c_.getY() + c_.getX() *
d_.getY() + d_.getX() * e_.getY() + e_.getX() * f_.getY() +
     - (b_.getX() * a_.getY() + c_.getX() * b_.getY() +
    d_.getX() * c_.getY() + e_.getX() * d_.getY() + f_.getX() * e_.getY())));
}

void Hexagon::Print(std::ostream& ssd)
{
    std::cout << "Hexagon: " << a_ << " " << b_ << " ";
    std::cout << c_ << " " << d_ << " " << e_ << " ";
    std::cout << f_ << "\n";
}
```

octagon.h

```
#ifndef OCTAGON_H
#define OCTAGON_H

#include "figure.h"

class Octagon : Figure
{
public:
   Octagon(std::istream& ins);
   size_t VertexesNumber();
   double Area();
   void Print(std::ostream& ssd);

private:
   Point a_, b_, c_, d_;
   Point e_, f_, g_, h_;
};

#endif
```

octagon.cpp

```
#include "octagon.h"

Octagon::Octagon(std::istream& ins)
{
   std::cin >> a_ >> b_ >> c_ >> d_;
   std::cin >> e_ >> f_ >> g_ >> h_;
}

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

double Octagon::Area()
{
   return 0.5 * abs((a_.getX() * b_.getY() + b_.getX() * c_.getY() + c_.getX() * d_.getY() + d_.getX() *
e_.getY() + e_.getX() * f_.getY() +
   f_.getX() * g_.getY() + g_.getX() * h_.getY() + h_.getX() * a_.getY() - (b_.getX() * a_.getY() +
c_.getX() * b_.getY() +
   d_.getX() * c_.getY() + e_.getX() * d_.getY() + f_.getX() * e_.getY() + g_.getX() * f_.getY() +
h_.getX() * g_.getY() +
   a_.getX() * h_.getY())));
}

void Octagon::Print(std::ostream& ssd)
{
   std::cout << "Octagon: " << a_ << " " << b_ << " ";
   std::cout << c_ << " " << d_ << " " << e_ << " ";
   std::cout << f_ << " " << g_ << " " << h_ << "\n";
}
```

triangle.h

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


class Triangle: public Figure {
private:
    Point a_, b_, c_;
public:
    Triangle();
    Triangle(const Triangle& triangle);
    Triangle(std::istream &is);
    size_t VertexesNumber();
    double Area();
    void Print(std::ostream& os);

};


#endif
```

triangle.cpp

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

Triangle::Triangle() : a_(0,0), b_(0,0), c_(0,0)
{}

Triangle::Triangle(const Triangle& triangle)
{
    this->a_ = triangle.a_;
    this->b_ = triangle.b_;
    this->c_ = triangle.c_;
}

Triangle::Triangle(std::istream &is)
{
    std::cin >> a_ >> b_ >> c_;
}

size_t Triangle::VertexesNumber()
{
    return 3;
}

double Triangle::Area()
{
    double a = a_.dist(b_);
    double b = b_.dist(c_);
    double c = c_.dist(a_);
    double p = (a + b + c)/2;
    return sqrt(p*(p-a)*(p-b)*(p-c));
}

void Triangle::Print(std::ostream& os)
{
    std::cout << "Triangle " << a_ << b_ << c_ << std::endl;
```

```
}
```