МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

ЛАБОРАТОРНАЯ РАБОТА №3

по курсу “Объектно-ориентированное программирование”

I семестр, 2021/22 учебный год

Студент: *Мерц Савелий Павлович, группа М8О-207Б-20*

Преподаватель: *Дорохов Евгений Павлович, каф. 806*

Задание:

Необходимо спроектировать и запрограммировать на языке C++ классы трех фигур, согласно варианту задания.

Вариант №14:

- Фигура 1: Треугольник (Triangle)
- Фигура 2: Квадрат (Square)
- Фигура 3: Восьмиугольник (Octagon)

Описание программы:

Исходный код разделен на 10 файлов:

- point.h – описание класса точки
- point.cpp – реализация класса точки
- figure.h – описание класса фигуры
- octagon.h – описание класса восьмиугольника (наследуется от фигуры)
- octagon.cpp – реализация класса восьмиугольника
- tree_elem.h – описание элемента дерева
- tree_elem.cpp – реализация элемента дерева
- tbinarytree.h – описание дерева
- tbinarytree.cpp – реализация дерева
- main.cpp – основная программа

Дневник отладки:

Результат подсчёта площади формулой гаусса в моей реализации почему-то был равен нулю в для фигур с отличной от нуля площадью. Чтобы реализовать метод Area мне пришлось использовать подсчет через площади треугольников, образующихся из восьмиугольника и одной внутренней точки. Получилось дробное число вместо целого. Такой результат меня не устраивал. Оказалось, что в геттере позиции вершины восьмиугольника, необходимого для формулы гаусса, я возвращал точку с координатами (x,x) вместо (x,y). После соответствующих исправлений площадь подсчитывалась должным образом. Данное исправление сделало бесполезным пару методов, которые были удалены из программы.

Вывод:

При выполнении работы я на практике познакомился с базовыми принципами ООП, реализовал несколько классов, конструкторы и методы для работы с ними. Также

я перегрузил операторы ввода/вывода для удобной работы с реализованными классами. Использовать спецификацию как форму наследования очень удобно при реализации подклассов, для каждого из которых необходимо соблюдать одинаковое поведение.

Исходный код:

point.h:

```
#ifndef POINT_H
#define POINT_H

#include <iostream>

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

  double getX();
  double getY();

  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::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;
}
```

figure.h:

```
#ifndef FIGURE_H
#define FIGURE_H

#include <iostream>

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

#endif // FIGURE_H
```

octagon.h:

```
#ifndef OCTAGON_H
#define OCTAGON_H

#include <iostream>

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


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();

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

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

#endif
```

octagon.cpp:

```
#include "octagon.h"
```

```
#include <cmath>

Octagon::Octagon()
    : a_(0, 0),
     b_(0, 0),
     c_(0, 0),
     d_(0, 0),
     e_(0, 0),
     f_(0, 0),
     g_(0, 0),
     h_(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) {
    is >> a_ >> b_ >> c_ >> d_ >> e_ >> f_ >> g_ >> 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;

}

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

void Octagon::Print(std::ostream& os) {
    os << "Octagon: ";
    os <<  a_ << " ";
    os <<  b_ << " ";
    os <<  c_ << " ";
    os <<  d_ << " ";
    os <<  e_ << " ";
    os <<  f_ << " ";
    os <<  g_ << " ";
    os <<  h_ << " ";
    os << std::endl;
}

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()
       -   a_.getY()*b_.getX()   -   b_.getY()*c_.getX()   -   c_.getY()*d_.getX()   -
d_.getY()*e_.getX()     -     e_.getY()*f_.getX()     -     f_.getY()*g_.getX()     -
g_.getY()*h_.getX() - h_.getY()*a_.getX());
}

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

triangle.h:

```
#ifndef TRIANGLE_H
#define TRIANGLE_H

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

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

    virtual ~Triangle();

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

private:
    double len_a, len_b, len_c;
};

#endif
```

triangle.cpp:

```
#include "triangle.h"

#include <cmath>

static Point a_, b_, c_;

Triangle::Triangle()
    : len_a(0.0),
      len_b(0.0),
      len_c(0.0) {
  //std::cout << "Default triangle created" << std::endl;
}

Triangle::Triangle(double a, double b, double c)
    : len_a(a),
```

```
      len_b(b),
      len_c(c) {
  //std::cout << "Triangle created" << std::endl;
}

Triangle::Triangle(std::istream &is) {
    is >> a_ >> b_ >> c_;
      len_a = a_.dist(b_);
      len_b = b_.dist(c_);
      len_c = c_.dist(a_);
  //std::cout << "Triangle created via istream" << std::endl;
}

Triangle::Triangle(const Triangle& other)
    : Triangle(other.len_a, other.len_b, other.len_c) {
  //std::cout << "Made copy of triangle" << std::endl;
}

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

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

double Triangle::Area() {
    double p = (len_a + len_b + len_c) / 2.0;
    return sqrt(abs(p * (p - len_a) * (p - len_b) * (p - len_c)));
}

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

square.h:

```
#ifndef SQUARE_H
#define SQUARE_H

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


class Square : public Figure {
public:
    Square();
    Square(double a, double b, double c, double d);
    Square(std::istream &is);
    Square(const Square& other);

    virtual ~Square();

    size_t VertexesNumber();
```

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

private:
    double len_ab, len_bc, len_cd, len_da;
};

#endif
```

square.cpp:

```
#include "square.h"

#include <cmath>

static Point a_, b_, c_, d_;

Square::Square()
    : len_ab(0.0),
      len_bc(0.0),
      len_cd(0.0),
      len_da(0.0) {
  //std::cout << "Default square created" << std::endl;
}

Square::Square(double a, double b, double c, double d)
    : len_ab(a),
      len_bc(b),
      len_cd(c),
      len_da(d) {
  //std::cout << "Square created" << std::endl;
}

Square::Square(std::istream &is) {
    is >> a_ >> b_ >> c_ >> d_;
      len_ab = a_.dist(b_);
      len_bc = b_.dist(c_);
      len_cd = c_.dist(d_);
      len_da = d_.dist(a_);
  //std::cout << "Square created via istream" << std::endl;
}

Square::Square(const Square& other)
    : Square(other.len_ab, other.len_bc, other.len_cd, other.len_da) {
  //std::cout << "Made copy of square" << std::endl;

}

size_t Square::VertexesNumber() {
    return 4;
}

void Square::Print(std::ostream& os) {
    //std::ostream& os = std::cout;
    os << "Square: ";
    os <<  a_ << " ";
    os << b_ << " ";
    os << c_ << " ";
```

```
    os << d_ <<  " ";
    os << std::endl;
}

double Square::Area() {
    return len_ab * len_bc;
}

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

main.cpp:

```
#include <iostream>
#include "triangle.h"
#include "square.h"
#include "octagon.h"
#include "point.h"


int main()
{
    std::cout << "Enter the points' coordinates of triangle:\n";
    Triangle a(std::cin);
    std::cout << "Triangle's number of vertexes: " << a.VertexesNumber() << "\n";
    std::cout << "Triangles's area: " << a.Area() << "\n";
    a.Print(std::cout);
    std::cout << std::endl;

    std::cout << "Enter the points' coordinates of square:\n";
    Square b(std::cin);
    std::cout << "Square's number of vertexes: " << b.VertexesNumber() << "\n";
    std::cout << "Square's area: " << b.Area() << "\n";
    b.Print(std::cout);
    std::cout << std::endl;

    std::cout << "Enter the points' coordinates of octagon:\n";
    Octagon c(std::cin);
    std::cout << "Octagon's number of vertexes: " << c.VertexesNumber() << "\n";
    std::cout << "Octagon's area: " << c.Area() << "\n";
    c.Print(std::cout);
    std::cout << std::endl;

    return 0;
}
```