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

ЛАБОРАТОРНАЯ РАБОТА №3

по курсу "Объектно-ориентированное программирование" І 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 Агеа 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.

Вывол:

При выполнении работы я на практике познакомился с базывыми 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:

is >> p.x_ >> p.y_;

```
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);</pre>
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) {
```

```
return is;
std::ostream& operator<<(std::ostream& os, Point& p) {</pre>
 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, 0)
     e_{0}, 0),
     f_(0, 0),
     g_{0}(0, 0)
     h_(0, 0) {
  //std::cout << "Default octagon created" << std::endl;</pre>
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}(d),
     e_(e),
     f_{(f)}
     g_{g}(g),
     h_(h) {
  //std::cout << "Octagon created" << std::endl;</pre>
Octagon::Octagon(std::istream &is) {
    is \rightarrow a_ \rightarrow b_ \rightarrow c_ \rightarrow d_ \rightarrow e_ \rightarrow f_ \rightarrow g_ \rightarrow h_;
  //std::cout << "Octagon created via istream" << std::endl;</pre>
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;</pre>
}
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;</pre>
}
double Octagon::Area() {
```

```
return 0.5 * abs( a_.getX()*b_.getY() + b_.getX()*c_.getY() + d_.getY() + e_.getX()*f_.getY() +
c_.getX()*d_.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()
Y()*e_.getX() - e_.getY()*f_.getX() - f_.getY()*g_.getX()
d_.getY()*e_.getX()
g_.getY()*h_.getX() - h_.getY()*a_.getX());
Octagon::~Octagon() {
    //std::cout << "Octagon deleted" << std::endl;</pre>
}
       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;</pre>
Triangle::Triangle(double a, double b, double c)
    : len_a(a),
```

```
len_b(b),
      len_c(c) {
  //std::cout << "Triangle created" << std::endl;</pre>
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;</pre>
Triangle::Triangle(const Triangle& other)
    : Triangle(other.len_a, other.len_b, other.len_c) {
  //std::cout << "Made copy of triangle" << std::endl;</pre>
size_t Triangle::VertexesNumber() {
    return 3;
void Triangle::Print(std::ostream& os) {
    os << "Triangle: ";
os << a_ << " ";
    os << b_ << " ";
    os << c_ << " ";
    os << std::endl;</pre>
}
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;</pre>
}
      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;</pre>
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;</pre>
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;</pre>
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;</pre>
}
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;</pre>
}
double Square::Area() {
    return len_ab * len_bc;
Square::~Square() {
    //std::cout << "Square deleted" << std::endl;</pre>
}
       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";</pre>
    Triangle a(std::cin);
    std::cout << "Triangle's number of vertexes: " << a.VertexesNumber() << "\n";</pre>
    std::cout << "Triangles's area: " << a.Area() << "\n";</pre>
    a.Print(std::cout);
    std::cout << std::endl;</pre>
    std::cout << "Enter the points' coordinates of square:\n";</pre>
    Square b(std::cin);
    std::cout << "Square's number of vertexes: " << b.VertexesNumber() << "\n";</pre>
    std::cout << "Square's area: " << b.Area() << "\n";</pre>
    b.Print(std::cout);
    std::cout << std::endl;</pre>
    std::cout << "Enter the points' coordinates of octagon:\n";</pre>
    Octagon c(std::cin);
    std::cout << "Octagon's number of vertexes: " << c.VertexesNumber() << "\n";</pre>
    std::cout << "Octagon's area: " << c.Area() << "\n";</pre>
    c.Print(std::cout);
    std::cout << std::endl;</pre>
    return 0;
}
```