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

ЛАБОРАТОРНАЯ РАБОТА №3 по курсу

объектно-ориентированное программирование I семестр, 2021/22 уч. год

Студент *Михеева Кристина Олеговна, группа М8О-207Б-20*
Преподаватель *Дорохов Евгений Павлович*

Условие

Вариант 17: Треугольник, Квадрат, Прямоугольник. 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 расположены 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 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 команд 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/rectangle.h: описание класса прямоугольника, наследующегося от figures .
6. include/square.h: описание класса квадрата, наследующегося от rectangle.
7. include/point.cpp: реализация класса точки
8. include/triangle.cpp: реализация класса треугольника, наследующегося от figures
9. include/rectangle.cpp: реализация класса прямоугольника, наследующегося от figures
10. include/square.cpp: реализация класса квадрата, наследующегося от rectangle

Дневник отладки:

При выполнении данной работы возникли некоторые проблемы с утечкой памяти, которые были исправлены.

Выводы:

В данной лабораторной работе я познакомилась с полиформизмом и наследованием. Достичь этого получилось при помощи реализации класса “Figure”.Ведь именно от этого класса наследуются предложенные фигуры(в моем случае: треугольник, квадрат и прямоугольник). А полиморфизм достигается за счет виртуальных функций, где ключевым слово является virtual. Описав виртуальные методы “Print, Area, VertexesNumber**”,** мы автоматически реализовали себе данные методы в каждом классе многоугольников по-разному. В этом и заключается принцип полиморфизма в данной работе.

Листинг

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 &os) = 0;
    virtual ~Figure() {};
};

#endif
```

point.h

```
#ifndef POINT_H
#define POINT_H

#include <iostream>

class Point {
public:
  Point();
  Point(std::istream &is);
  Point(double x, double y);
  friend std:: istream& operator>>(std:: istream& is, Point& p);
  friend std:: ostream& operator<<(std:: ostream& os, Point& p);

double getX();
```

```
double getY();

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_;
}

std::istream& operator>>(std:: istream& is, Point& p) {
  is >> p.x_ >> p.y_;
  return is;
}

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

double Point:: getY() {
    return y_;
};

std:: ostream& operator<<(std:: ostream& os, Point& p) {
  os << "(" << p.x_ << ", " << p.y_ << ")";
  return os;
}
```

triangle.h

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

class Triangle : public Figure {
    public:
    Triangle(std:: istream &is);
    size_t VertexesNumber();
    double Area();
    void Print (std:: ostream &os);
    virtual ~Triangle();
```

```
    private:
    Point a;
    Point b;
    Point c;
};
```

triangle.cpp

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

Triangle::Triangle(std:: istream &is)
{
    is >> a >> b >> c;
    std:: cout << "The triangle was created" << std:: endl;
}

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

double Triangle::Area() {
    double Square = 0.5 * abs(a.getX() * b.getY() + b.getX() * c.getY() +
c.getX() * a.getY() - a.getY() * b.getX() - b.getY() * c.getX() - c.getY() *
a.getX());
    return Square;
}

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

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

square.h

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

class Square : public Figure {
    public:
    Square(std:: istream &is);
    size_t VertexesNumber();
    double Area();
    void Print (std:: ostream &os);
    virtual ~Square();
```

```
    private:
    Point a;
    Point b;
    Point c;
    Point d;
};
```

square.cpp

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

Square::Square (std:: istream &is)
{
    is >> a >> b >> c >> d;
    std:: cout << "The square created" << std:: endl;
}

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

double Square::Area() {
    double SquareS =
abs(a.getX()*b.getY()+b.getX()*c.getY()+c.getX()*d.getY()+d.getX()*a.getY() -
a.getY()*b.getX() - b.getY()*c.getX() - c.getY()*d.getX() -
d.getY()*a.getX());
    return SquareS;
}

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

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

rectangle.h

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

class Rectangle : public Figure {
    public:
    Rectangle(std:: istream &is);
    size_t VertexesNumber();
```

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

    private:
    Point a;
    Point b;
    Point c;
    Point d;
};
```

rectangle.cpp

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

Rectangle::Rectangle (std:: istream &is)
{
    is >> a >> b >> c >> d;
        std:: cout << "The rectagon was created" << std:: endl;
}

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

double Rectangle::Area() {
    double RectangleS =
abs(a.getX()*b.getY()+b.getX()*c.getY()+c.getX()*d.getY()+d.getX()*a.getY() -
a.getY()*b.getX() - b.getY()*c.getX() - c.getY()*d.getX() -
d.getY()*a.getX());
    return RectangleS;
}

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

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

main.cpp

```
#include "figure.h"
#include "triangle.h"
#include "rectangle.h"
#include "square.h"
#include <iostream>
```

```
int main () {
    Triangle a (std:: cin);
    std:: cout << "Number of vertices in a triangle:" << " " <<
a.VertexesNumber() << std:: endl;
    a.Print(std:: cout);
    std:: cout << "Area of a triangle:" << " " << a.Area() << std:: endl;

    Square b (std:: cin);
    std:: cout << "Number of vertices for a  square: " << " "<<
b.VertexesNumber() << std:: endl;
    b.Print(std:: cout);
    std:: cout << "Square area:" <<" " << b.Area() << std:: endl;


    Rectangle c (std:: cin);
    std:: cout << "Number of vertices for a rectangle:"<< " " <<
c.VertexesNumber() << std:: endl;
    c.Print(std:: cout);
    std:: cout << "Rectangle area:"<< " " << c.Area() << std:: endl;

    return 0;
```

3