

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: Меджидли Махмуд Ибрагим оглы, группа М80-208Б-20

Преподаватель: Дорохов Евгений Павлович, каф. 806

Задание:

Спроектировать и запрограммировать на языке C++ классы трёх фигур. Классы должны удовлетворять следующим правилам:

- Должны быть названы как в вариантах задания и расположены в отдельных файлах;
- Иметь общий родительский класс Figure;
- Содержать конструктор, принимающий координаты вершин фигуры из стандартного потока `std::cin`, расположенных через пробел (например: `0.0 0.0 1.0 0.0 1.0 1.0 0.0 1.0`);
- Содержать набор общих методов:
 - `size_t VertexesNumber()` – метод, возвращающий количество вершин фигуры
 - `double Area()` – метод расчета площади фигуры

Вариант №12:

- Фигура 1: Пятиугольник (Pentagon)
- Фигура 2: Ромб (Rhombus)
- Фигура 3: Трапеция (Trapezoid)

Описание программы:

Исходный код разделён на 10 файлов:

- `point.h` – описание класса точки
- `point.cpp` – реализация класса точки
- `figure.h` – описание класса фигуры
- `pentagon.h` – описание класса пятиугольника
- `pentagon.cpp` – реализация класса пятиугольника
- `trapezoid.h` – описание класса трапеции
- `trapezoid.cpp` – реализация класса трапеции
- `rhombus.h` – описание класса ромба
- `rhombus.cpp` – реализация класса ромба
- `main.cpp` – основная программа

Дневник отладки:

Программа в отладке не нуждалась, необходимый функционал был реализован довольно быстро и безошибочно.

Вывод:

Благодаря ЛР №3 я знаю, что такое полиморфизм и наследование. Достигается это при помощи реализации класса “Figure”. От этого класса далее наследуются наши пятиугольники, трапеции и ромбы. А полиморфизм достигается за счет виртуальных функций (**virtual**). Описав виртуальные методы **Print**, **Square**, **VertexesNumber**, мы автоматически позволили сами же себе реализовать эти методы в каждом классе многоугольников по-разному. В этом и заключается принцип полиморфизма в данной ЛР.

Исходный код:

point.h:

```
#ifndef POINT_H
#define POINT_H
#include <iostream>
#include <cmath>

class Point
{
public:
    Point();
    Point(std::istream& is);
    Point(double x, double y);
    double length(Point& p1, Point& p2);
    friend std::istream& operator>>(std::istream& is, Point& p);
    friend std::ostream& operator<<(std::ostream& os, Point& p);
    friend double dist(Point& p1, Point& p2);

private:
    double x_, y_;
};

#endif
```

point.cpp:

```
#include "point.h"

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 dist(Point& p1, Point& p2)
```

```
{  
    double dx = (p1.x_ - p2.x_);  
    double dy = (p1.y_ - p2.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;  
}
```

figure.h:

```
#ifndef FIGURE_H  
#define FIGURE_H  
#include "point.h"  
  
class Figure  
{  
public:  
    virtual void Print(std::ostream& os) = 0;  
    virtual double Square() = 0;  
    virtual ~Figure() {};  
    virtual size_t VertexesNumber() = 0;  
};  
  
#endif
```

pentagon.h:

```

#ifndef PENTAGON_H
#define PENTAGON_H
#include "figure.h"

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

private:
    Point a, b, c, d, e;
    double len1, len2, len3, len4, len5, diag1, diag2;
};

#endif

```

pentagon.cpp:

```

#include "pentagon.h"
#include <math.h>
Pentagon::Pentagon() : a(0.0, 0.0), b(0.0, 0.0), c(0.0, 0.0), d(0.0, 0.0), e(0.0, 0.0)
{
    std::cout << "Created default pentagon" << std::endl;
};

Pentagon::Pentagon(std::istream& is)
{
    std::cout << "Enter the coordinates of pentagon's points (x and y)" << std::endl;
    std::cout << "First enter lower left vertex and then go clockwise" << std::endl;
    is >> a >> b >> c >> d >> e;
    len1 = dist(a, b);
    len2 = dist(b, c);
    len3 = dist(c, d);
    len4 = dist(d, e);
    len5 = dist(e, a);
    diag1 = dist(a, c);
    diag2 = dist(c, e);
    std::cout << "Created pentagon via istream" << std::endl;
}

double Pentagon::Square() {
    double p1 = (len1 + len2 + diag1) / 2;
    double s1 = sqrt(p1 * (p1 - len1) * (p1 - len2) * (p1 - diag1));
    double p2 = (diag1 + diag2 + len5) / 2;
    double s2 = sqrt(p2 * (p2 - diag1) * (p2 - diag2) * (p2 - len5));
    double p3 = (diag2 + len3 + len4) / 2;
    double s3 = sqrt(p3 * (p3 - diag2) * (p3 - len3) * (p3 - len4));
    double square = s1+s2+s3;
    return square;
}

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

```

```

size_t Pentagon::VertexesNumber()
{
    return 5;
}

Pentagon::~Pentagon()
{
    std::cout << "Deleted pentagon" << std::endl;
}

```

trapezoid.h:

```

#ifndef TRAPEZOID_H
#define TRAPEZOID_H
#include "figure.h"
#include <algorithm>
class Trapezoid : public Figure
{
public:
    Trapezoid();
    Trapezoid(std::istream& is);
    virtual ~Trapezoid();
    void Print(std::ostream& os);
    double Square();
    size_t VertexesNumber();

private:
    Point a, b, c, d;
    double lena, lenb, lenc, lend;
};

#endif

```

trapezoid.cpp:

```

#include "trapezoid.h"

Trapezoid::Trapezoid() : a(0.0, 0.0), b(0.0, 0.0), c(0.0, 0.0), d(0.0, 0.0)
{
    std::cout << "Created default trapezoid" << std::endl;
};

Trapezoid::Trapezoid(std::istream& is)
{
    std::cout << "Enter the coordinates of trapezoid's points (x and y)" << std::endl;
    std::cout << "First enter top left vertex and then go clockwise" << std::endl;
    is >> a >> b >> c >> d;
    lena = dist(a, b);
    lenb = dist(c, d);
    lenc = dist(b, c);
    lend = dist(a, d);
    if (lena > lenb)
    {
        std::swap(lena, lenb);
        std::swap(lenc, lend);
    }
    std::cout << "Created trapezoid via istream" << std::endl;
}

```

```

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

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

double Trapezoid::Square()
{
    return ((lena + lenb) / 2) * sqrt(pow(lenc, 2) - pow(((pow(lenb - lena, 2) + pow(lenc, 2) - pow(lend, 2)) / (2. * (lenb - lena))), 2));
}

Trapezoid::~Trapezoid()
{
    std::cout << "Deleted trapezoid" << std::endl;
}

```

rhombus.h:

```

#ifndef RHOMBUS_H
#define RHOMBUS_H
#include "figure.h"

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

private:
    Point a, b, c, d;
    double diag1, diag2;
};

#endif

```

rhombus.cpp:

```

#include "rhombus.h"

Rhombus::Rhombus() : a(0.0, 0.0), b(0.0, 0.0), c(0.0, 0.0), d(0.0, 0.0)
{
    std::cout << "Created default rhombus" << std::endl;
};

Rhombus::Rhombus(std::istream& is)
{
    std::cout << "Enter the coordinates of pentagon's points (x and y)" << std::endl;
    std::cout << "First enter left vertex and then go clockwise" << std::endl;
    is >> a >> b >> c >> d;
}

```

```

        diag1 = dist(a, c);
        diag2 = dist(b, d);
        std::cout << "Created rhombus via istream" << std::endl;
    }

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

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

double Rhombus::Square()
{
    return (diag1 * diag2) / 2.;
}

Rhombus::~Rhombus()
{
    std::cout << "Deleted rhombus" << std::endl;
}

```

main.cpp

```

#include "pentagon.h"
#include "trapezoid.h"
#include "rhombus.h"
int main()
{
    Pentagon rec1(std::cin);
    rec1.Print(std::cout);
    std::cout << rec1.VertexesNumber() << std::endl;
    std::cout << rec1.Square() << std::endl;

    Trapezoid t1(std::cin);
    t1.Print(std::cout);
    std::cout << t1.VertexesNumber() << std::endl;
    std::cout << t1.Square() << std::endl;

    Rhombus r1(std::cin);
    r1.Print(std::cout);
    std::cout << r1.VertexesNumber() << std::endl;
    std::cout << r1.Square() << std::endl;

    Figure* rec2 = new Pentagon(std::cin);
    rec2->Print(std::cout);
    std::cout << rec2->VertexesNumber() << std::endl;
    std::cout << rec2->Square() << std::endl;
    delete rec2;

    Figure* t2 = new Trapezoid(std::cin);
    t2->Print(std::cout);
    std::cout << t2->VertexesNumber() << std::endl;
    std::cout << t2->Square() << std::endl;
    delete t2;
}

```



```
Figure* r2 = new Rhombus(std::cin);
r2->Print(std::cout);
std::cout << r2->VertexesNumber() << std::endl;
std::cout << r2->Square() << std::endl;
delete r2;

system("pause");
return 0;
}
```