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

ЛАБОРАТОРНАЯ РАБОТА №3

по курсу “Объектно-ориентированное программирование» 1 семестр,

2021/22 уч. год

Студентка: *Колпакова Диана Саргаевна, группа М8О-208Б-20*
Преподаватель: *Дорохов Евгений Павлович*

Задание

Необходимо спроектировать и запрограммировать на языке C++ классы трех фигур, согласно варианту задания. Классы должны удовлетворять следующим правилам:

- Должны быть названы так же, как в вариантах задания, и расположены в раздельных файлах: отдельно заголовки (имя_класса_с_маленькой_буквы.h), отдельно описания методов (имя_класса_с_маленькой_буквы.cpp);
- Иметь общий родительский класс Figure;
- Содержать конструктор по умолчанию;
- Содержать конструктор, принимающий координаты вершин фигуры из стандартного потока std::cin, расположенных через пробел. Пример: 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;
  - 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)\n

Программа должна позволять:

- Вводить произвольные фигуры и добавлять их в общий контейнер. Разрешается использовать стандартные контейнеры std;
- Распечатывать содержимое контейнера.

Вариант 9:

Фигура №1	**Имя класса**	**Фигура №2**	**Имя класса**	**Фигура №3**	**Имя класса**
Треугольник	Triangle	Квадрат	Square	Прямоугольник	Rectangle

Описание программы

Исходный код лежит в 11 файлах:

1. main.cpp: часть программы, отвечающая за взаимодействие с пользователем через консоль. В ней происходит инициализация объектов и вызов функций работы с ними, заполнение стандартного контейнера вектор введенными объектами и печать его содержимого;
2. point.h: описание класса Point точек A(a1, a2);
3. point.cpp: реализация класса Point;
4. figure.h: описание абстрактного класса-родителя Figure;
5. figure.cpp: реализация класса Figure;
6. triangle.h: описание класса Triangle треугольников, заданных по трем точкам, наследника Figure;
7. triangle.cpp: реализация класса Triangle;
8. rectangle.h: описание класса Rectangle, наследника Figure (объект – прямоугольник, заданный по точкам);
9. rectangle.cpp: реализация класса Rectangle;
10. square.h: описание класса Square, наследника Figure (объект – квадрат, заданный по точкам);
11. square.cpp: реализация класса Square.

Также используется файл CMakeLists.txt с конфигурацией CMake для автоматизации сборки программы.

Дневник отладки

Проблем не возникло

Вывод

В данной лабораторной работе я продолжила знакомиться с основами ООП в языке C++. Было изучено понятие абстрактного класса и виртуальной функции. Кроме принципов абстракции и инкапсуляции, были применены на практике принципы наследования и полиморфизма.

Базовым классом в наследовании стал класс Figure, от которого дочерние Triangle, Rectangle и Square унаследовали общие имена методов VertexesNumber,

Area, Print со своей реализацией для каждого класса, что возможно в C++ благодаря полиморфизму.

Таким образом, в данной работе я получила важные теоретические навыки для работы с объектно-ориентированными языками, а также расширила свои знания о программировании в C++.

Исходный код

main.cpp:

```
#include <iostream>
#include <vector>

#include "triangle.h"
#include "rectangle.h"
#include "square.h"

int main() {
  std::vector<Figure*> figures;

  std::cout << "Enter triangle vertices' coordinates: "; // -4 2 2 0 0 -2
  Triangle t1(std::cin);
  std::cout << "Number of vertices: " << t1.VertexesNumber() << std::endl;
  t1.Print(std::cout);
  std::cout << "Area:" << t1.Area() << std::endl;
  figures.push_back(&t1);

  std::cout << "Enter rectangle vertices' coordinates: "; // 0 0 5 0 5 -2 0 -2
  Rectangle r1(std::cin);
  std::cout << "Number of vertices: " << r1.VertexesNumber() << std::endl;
```

```
  r1.Print(std::cout);
  std::cout << "Area:" << r1.Area() << std::endl;
  figures.push_back(&r1);

  std::cout << "Enter square vertices' coordinates: "; // -1 1 1 1 1 -1 -1 -1
  Square s1(std::cin);
  std::cout << "Number of vertices: " << s1.VertexesNumber() << std::endl;
  s1.Print(std::cout);
  std::cout << "Area:" << s1.Area() << std::endl;
  figures.push_back(&s1);

  std::cout << "Enter rectangle vertices' coordinates: "; // -2.5 3 2.5 3 2.5 -
3 0 -3
  Rectangle r2(std::cin);
  std::cout << "Number of vertices: " << r2.VertexesNumber() << std::endl;
  r2.Print(std::cout);
  std::cout << "Area:" << r2.Area() << std::endl;
  figures.push_back(&r2);

  std::cout << "\nFigures in container:" << std::endl;
  for (int i = 0; i < figures.size(); i++)
  {
    figures[i]->Print(std::cout);
  }

  return 0;
}
```

point.h:

```
#ifndef POINT_H
```

```
#define POINT_H

#include <iostream>

class Point {

friend std::istream& operator>>(std::istream& is, Point& p);
friend std::ostream& operator<<(std::ostream& os, Point& p);

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

  double dist(Point& other);

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::dist(Point& other) {
  double dx = (other.x_ - x_);
  double dy = (other.y_ - 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 size_t VertexesNumber() = 0;
  virtual void Print(std::ostream& os) = 0;
  virtual double Area() = 0;
  virtual ~Figure() {};
};

#endif // FIGURE_H
```

triangle.h:

```
#ifndef TRIANGLE_H
```

```
#define TRIANGLE_H

#include <iostream>

#include "figure.h"

class Triangle : public Figure {

public:
  Triangle();
  Triangle(Point a, Point b, Point c);
  Triangle(std::istream &is);
  Triangle(const Triangle& other);

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

  virtual ~Triangle();

private:
  Point p1;
  Point p2;
  Point p3;
};

#endif // TRIANGLE_H
```

triangle.cpp:

```
#include "triangle.h"

#include <iostream>
#include <cmath>

Triangle::Triangle()
    : p1(0.0, 0.0), p2(0.0, 0.0), p3(0.0, 0.0) {  // можно, но длиннее
p1(Point(0.0, 0.0))
  std::cout << "Default triangle created" << std::endl;
}
```

```
Triangle::Triangle(Point a, Point b, Point c)
    : p1(a), p2(b), p3(c) {
  std::cout << "Triangle created by parameters" << std::endl;
}

Triangle::Triangle(std::istream &is) {
  is >> p1 >> p2 >> p3;
}

Triangle::Triangle(const Triangle& other)
    : Triangle(other.p1, other.p2, other.p3) {
  std::cout << "Triangle copy created" << std::endl;
}

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

double Triangle::Area() {
  double p12 = p1.dist(p2);
  double p13 = p1.dist(p3);
  double p23 = p2.dist(p3);
  double p = (p12 + p23 + p13) / 2.0;
  return std::sqrt(p * (p - p12) * (p - p23) * (p - p13));
}

void Triangle::Print(std::ostream& os) {
  os << "Triangle: ";
  os << p1 << ", ";
  os << p2 << ", ";
  os << p3 << std::endl;
}

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

square.h:

```
#ifndef SQUARE_H
#define SQUARE_H

#include <iostream>

#include "figure.h"

class Square : public Figure {

public:
  Square();
  Square(Point a, Point b, Point c, Point d);
  Square(std::istream &is);
  Square(const Square& other);

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

  virtual ~Square();

private:
  Point p1;
  Point p2;
  Point p3;
  Point p4;
};

#endif // SQUARE_H
```

square.cpp:

```
#include "square.h"

#include <iostream>
#include <cmath>

Square::Square()
    : p1(0.0, 0.0), p2(0.0, 0.0), p3(0.0, 0.0), p4(0.0, 0.0) {
  std::cout << "Default square created" << std::endl;
```

```
}

Square::Square(Point a, Point b, Point c, Point d)
    : p1(a), p2(b), p3(c), p4(d) {
  std::cout << "Square created by parameters" << std::endl;
}

Square::Square(std::istream &is) {
  is >> p1 >> p2 >> p3 >> p4;
}

Square::Square(const Square& other)
    : Square(other.p1, other.p2, other.p3, other.p4) {
  std::cout << "Square copy created" << std::endl;
}

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

double Square::Area() {
  double p12 = p1.dist(p2);
  double p = p12 * p12;
  return p;
}

void Square::Print(std::ostream& os) {
  os << "Square: ";
  os << p1 << ", ";
  os << p2 << ", ";
  os << p3 << ", ";
  os << p4 << std::endl;
}

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

rectangle.h:

```
#ifndef RECTANGLE_H
#define RECTANGLE_H

#include <iostream>

#include "figure.h"

class Rectangle : public Figure {

public:
  Rectangle();
  Rectangle(Point a, Point b, Point c, Point d);
  Rectangle(std::istream &is);
  Rectangle(const Rectangle& other);

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

  virtual ~Rectangle();

private:
  Point p1;
  Point p2;
  Point p3;
  Point p4;
};

#endif // RECTANGLE_H
```

rectangle.cpp:

```
#include "rectangle.h"

#include <iostream>
#include <cmath>

Rectangle::Rectangle()
    : p1(0.0, 0.0), p2(0.0, 0.0), p3(0.0, 0.0), p4(0.0, 0.0) {
  std::cout << "Default rectangle created" << std::endl;
```

```
}

Rectangle::Rectangle(Point a, Point b, Point c, Point d)
    : p1(a), p2(b), p3(c), p4(d) {
  std::cout << "Rectangle created by parameters" << std::endl;
}

Rectangle::Rectangle(std::istream &is) {
  is >> p1 >> p2 >> p3 >> p4;
}

Rectangle::Rectangle(const Rectangle& other)
    : Rectangle(other.p1, other.p2, other.p3, other.p4) {
  std::cout << "Rectangle copy created" << std::endl;
}

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

double Rectangle::Area() {
  double p12 = p1.dist(p2);
  double p23 = p2.dist(p3);
  double p = p12 * p23;
  return p;
}

void Rectangle::Print(std::ostream& os) {
  os << "Rectangle: ";
  os << p1 << ", ";
  os << p2 << ", ";
  os << p3 << ", ";
  os << p4 << std::endl;
}

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

CMakeLists.txt:

```
cmake_minimum_required(VERSION 3.10)
project(lab1)

set(CMAKE_CXX_STANDARD 11)


add_executable(lab1 point.h
  point.cpp
  main.cpp
  figure.h
  triangle.h triangle.cpp
 rectangle.h rectangle.cpp square.h square.cpp)
```