МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

**(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСТИТЕТ)**

ЛАБОРАТОРНАЯ РАБОТА

№1

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

Студент *Марини Иван Сергеевич, группа М8О-208Б-20*

Преподаватель *Дорохов Евгений Павлович*

**Условие**

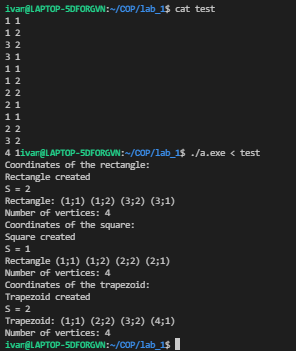
Задание: Вариант 10: Трапеция, Квадрат, Прямоугольник. Необходимо спроекти- ровать и запрограммировать на языке C++ классы трех фигур, согласно варианту задания. Классы должны удовлетворять следующим правилам:

1. Должны быть названы также, как в вариантах задания и расположенны в раздель- ных 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 комманд из меню
  2. include/figure.h: описание абстрактного класса фигур
  3. include/point.h: описание класса точки
  4. include/trapeoid.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/trapeoid.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**

**Недочеты**

Не выявлено.

Вывод

В ходе лабораторной работы я научился работать с классами на языке С++, познакомился с перегрузкой операторов и дружественными функциями, а также с операциями ввода-вывода из стандартных библиотек.

**Исходный код программы**

**figure.h**

#ifndef FIGURE\_H

#define FIGURE\_H

#include <iostream>

#include <cstddef>

#include <cmath>

#include "point.h"

using namespace std;

class Figure {

public:

virtual size\_t VertexesNumber() = 0;

virtual double Area() = 0;

virtual void Print() = 0;

};

#endif

**main.cpp**

#include "point.h"

#include "figure.h"

#include "rectangle.h"

#include "square.h"

#include "trapezoid.h"

int main() {

Rectangle rectangle(cin);

cout << "S = " << rectangle.Area() << endl;

rectangle.Print();

cout << "Number of vertices: " << rectangle.VertexesNumber() << endl;

Square square(cin);

cout << "S = " << square.Area() << endl;

square.Print();

cout << "Number of vertices: " << square.VertexesNumber() << endl;

Trapezoid tarapezoid(cin);

cout << "S = " << tarapezoid.Area() << endl;

tarapezoid.Print();

cout << "Number of vertices: " << tarapezoid.VertexesNumber() << endl;

return 0;

}

**point.cpp**

#include <cmath>

#include "point.h"

Point :: Point() : x\_(0.0), y\_(0.0) {}

Point :: Point(double x, double y) : x\_(x), y\_(y) {}

Point :: Point (istream &is) {

is >> x\_ >> y\_;

}

double Point :: x() {

return x\_;

}

double Point :: y() {

return y\_;

}

istream &operator>>(istream &is, Point &p) {

is >> p.x\_ >> p.y\_;

return is;

}

ostream &operator<<(ostream &os, Point &p) {

os << "(" << p.x\_ << ";" << p.y\_ << ")";

return os;

}

**point.h**

#ifndef POINT\_H

#define POINT\_H

#include <iostream>

using namespace std;

class Point {

public:

Point();

Point (istream &is);

Point(double x, double y);

double x();

double y();

friend istream &operator>>(istream &is, Point &p);

friend ostream &operator<<(ostream &os, Point&p);

friend class Square;

friend class Rectangle;

friend class Tarapezoid;

private:

double x\_;

double y\_;

};

#endif

**rectangle.cpp**

#include "point.h"

#include "figure.h"

#include "rectangle.h"

Rectangle :: Rectangle (istream &is) {

cout << "Coordinates of the rectangle: " << endl;

cin >> a >> b >> c >> d;

cout << "Rectangle created" << endl;

}

void Rectangle :: Print() {

cout << "Rectangle: " << a << ' ' << b << ' ' << c << ' ' << d << endl;

}

double Rectangle :: Area () {

double s = abs((a.x() - c.x()) \* (a.y() - b.y()));

return s;

}

size\_t Rectangle :: VertexesNumber() {

size\_t n = 4;

return n;

}

**rectangle.h**

#ifndef RECTANGLE\_H

#define RECTANGLE\_H

#include "figure.h"

class Rectangle : public Figure {

public:

Rectangle(istream &is);

void Print();

size\_t VertexesNumber();

double Area();

private:

Point a, b, c, d;

};

#endif