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 *Маринин Иван Сергеевич, группа М80-208Б-20*
Преподаватель *Дорохов Евгений Павлович*

Условие

Задание: Вариант 10: Трапеция, 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 расположенны 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 вершинв 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 файл лежит в 10 файлах:

1. main.cpp: основная программа, взаимодействие с пользователем посредством комманд из меню

2. figure.h: описание абстрактного класса фигур

3. point.h: описание класса точки
4. trapezoid.h: описание класса треугольника, наследующегося от figures
5. rectangle.h: описание класса прямоугольника, наследующегося от figures
6. square.h: описание класса квадрата, наследующегося от figures
7. point.cpp: реализация класса точки
8. trapezoid.cpp: реализация класса треугольника, наследующегося от figures
9. rectangle.cpp: реализация класса прямоугольника, наследующегося от figures
10. square.cpp: реализация класса квадрата, наследующегося от rectangle

Дневник отладки

Возникли небольшие проблемы при отладке программы. После тестирования они были устранены.

```
ivanmarinin@MacBook-Air-Ivan lab_1 % ./a.out
Coordinates of the rectangle:
1 1
1 2
3 2
3 1
Rectangle created
S = 2
Rectangle: (1;1) (1;2) (3;2) (3;1)
Number of vertices: 4
Coordinates of the square:
1 1
1 2
2 2
2 1
Square created
S = 1
```

```
Rectangle (1;1) (1;2) (2;2) (2;1)
Number of vertices: 4
Coordinates of the trapezoid:
1 1
2 2
3 2
4 1
Trapezoid created
S = 2
Trapezoid: (1;1) (2;2) (3;2) (4;1)
Number of vertices: 4
```

Недочёты

Недочётов не было обнаружено.

Выводы

В ходе лабораторной работы №3 я научился работать с классами на языке C++, познакомился с перегрузкой операторов и дружественными функциями, а также с операциями ввода-вывода из стандартных библиотек.

Исходный код

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;

}
```

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;
}
```

square.cpp

```
#include "point.h"
#include "figure.h"
#include "square.h"

Square :: Square (istream &is) {

    cout << "Coordinates of the square: " << endl;
    cin >> a >> b >> c >> d;
    cout  << "Square created" << endl;

}

void Square :: Print() {

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

}

double Square :: Area() {

    double s = pow(abs(a.x() - d.x()), 2);
    return s;

}

size_t Square :: VertexesNumber() {

    size_t n = 4;
    return n;

}
```

trapezoid.cpp

```
#include "point.h"
#include "figure.h"
#include "trapezoid.h"

Trapezoid :: Trapezoid (istream &is) {
```

```
    cout << "Coordinates of the trapezoid: " << endl;
    cin >> a >> b >> c >> d;
    cout << "Trapezoid created" << endl;
}

void Trapezoid :: Print() {

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

}

double Trapezoid :: Area() {

    double s = ((abs(a.x() - d.x()) + abs(b.x() - c.x())) / 2) *  abs(a.y() -
b.y());
    return s;

}

size_t Trapezoid :: VertexesNumber() {

    size_t n = 4;
    return n;

}
```

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;
}
```