Московский Авиационный Институт
(Национальный Исследовательский Университет)

Кафедра 806 «Вычислительная информатика и программирование»
Факультет: «Информационные технологии и прикладная математика»

Лабораторная работа
Дисциплина: «Объектно-ориентированное программирование»
I I семестр
Задание 1: «Простые классы»

| | |
|---|---|
| Группа: | М8О-108Б-18, №12 |
| Студент: | Коростелев Дмитрий Васильевич |
| Преподаватель: | Журавлёв Андрей Андреевич |
| Оценка: | |
| Дата: | 16.09.2019 |

Москва, 2019

## 1. Задание

(*вариант № 12*): Разработать класс `Rectangle`, представляющий собой прямоугольник со сторонами, параллельными осям координат. Поля – координаты левого нижнего и правого верхнего угла. Требуется реализовать следующие методы: вычисление площади и периметра, перемещения вдоль осей, изменение размеров, сравнение по площади и по периметру. Реализовать метод получения прямоугольника, представляющего общую часть (пересечение) двух прямоугольников. Реализовать метод объединения двух прямоугольников: наименьший прямоугольник, включающего оба заданных прямоугольника.

## 2. Адрес репозитория на GitHub

https://github.com/Dmitry4K/MAI_OOP/tree/master/variant%2012/lab1

## 3. Код программы на C++

main_io.cpp

```
#include <iostream>
#include <string>
#include <fstream>
#include "rectangle.h"

int main(int argc, char *argv[]){
    std::ifstream inFile(argv[1]);
    int a,b,c,d;
    inFile >> a >> b >> c >> d;
    Rectangle rec1 = Rectangle(a,b,c,d);
    inFile >> a >> b >> c >> d;
    Rectangle rec2 = Rectangle(a,b,c,d);
    inFile.close();
    std::string argument1 = argv[1];
    std::string name_file = "result_" + argument1;
    std::ofstream outFile;
    outFile.open(name_file);
    outFile << "First Rectangle:"<< std::endl;
    outFile << "      First Coordinate: X: "<< rec1.getFirstCoordinate().getX()<<" Y:
"<< rec1.getFirstCoordinate().getY()<<std::endl;
```

```
    outFile << "      Second Coordinate: X: "<<
rec1.getSecondCoordinate().getX()<<" Y: "<<
rec1.getSecondCoordinate().getY()<<std::endl;
    outFile << "      Square: "<< rec1.square()<< " Perimetr :
"<<rec1.perimetr()<<std::endl;

    outFile << "Second Rectangle:"<< std::endl;
    outFile << "      First Coordinate: X: "<< rec2.getFirstCoordinate().getX()<<" Y:
"<< rec2.getFirstCoordinate().getY()<<std::endl;
    outFile << "      Second Coordinate: X: "<<
rec1.getSecondCoordinate().getX()<<" Y: "<<
rec1.getSecondCoordinate().getY()<<std::endl;
    outFile << "      Square: "<< rec2.square()<< " Perimetr :
"<<rec2.perimetr()<<std::endl;

    outFile << "Compare on Square: " << rec1.compareSquare(rec2)<< " Compare
on Perimetr: " << rec1.comparePerimetr(rec2) << std::endl;
    outFile << "Intersection : "<<std::endl;
    Rectangle rec3 = rec1.intersection(rec2);
    outFile << "      First Coordinate: X: "<< rec3.getFirstCoordinate().getX()<<" Y:
"<< rec3.getFirstCoordinate().getY()<<std::endl;
    outFile << "      Second Coordinate: X: "<<
rec3.getSecondCoordinate().getX()<<" Y: "<<
rec3.getSecondCoordinate().getY()<<std::endl;
    outFile << "      Square: "<< rec3.square()<< " Perimetr :
"<<rec3.perimetr()<<std::endl;

    outFile << "Composition : "<<std::endl;
    rec3 = rec1.composition(rec2);
    outFile << "      First Coordinate: X: "<< rec3.getFirstCoordinate().getX()<<" Y:
"<< rec3.getFirstCoordinate().getY()<<std::endl;
    outFile << "      Second Coordinate: X: "<<
rec3.getSecondCoordinate().getX()<<" Y: "<<
rec3.getSecondCoordinate().getY()<<std::endl;
    outFile << "      Square: "<< rec3.square()<< " Perimetr :
"<<rec3.perimetr()<<std::endl;

    return 0;
}
```

Rectangle.h

```
#ifndef RECTANGLE_H
#define RECTANGLE_H
```

```
class Coordinate {
private:
    int x;
    int y;
public:
    Coordinate();
    Coordinate(int x, int y);
    int getX();
    int getY();
    void setX(int x);
    void setY(int y);
    void set(Coordinate f);
};

class Rectangle {
private:
    Coordinate first;
    Coordinate second;
public:
    Rectangle();
    Rectangle(Coordinate f, Coordinate s);
    Rectangle(int fX, int fY, int sX, int sY);
    Coordinate getFirstCoordinate();
    Coordinate getSecondCoordinate();
    int square();
    int perimetr();
    Rectangle intersection(Rectangle rec);
    Rectangle composition(Rectangle rec);
    void shift(int onX, int onY);
    void changeRectangle(Coordinate f, Coordinate s);
    bool compareSquare(Rectangle rec);
    bool comparePerimetr(Rectangle rec);
};

#endif

#include "rectangle.h"

int max(int a, int b){
    return a>b ? a : b;
}

int min(int a, int b){
    return a>b ? b : a;
}
```

```
int abs(int a){
    return a>0 ? a : -a;
}


Coordinate::Coordinate() {
    this->x = 0;
    this->y = 0;
}

Coordinate::Coordinate(int x, int y) {
    this->x = x;
    this->y = y;
}

int Coordinate::getX() {
    return x;
}
int Coordinate::getY() {
    return y;
}
void Coordinate::setX(int x) {
    this->x = x;
}
void Coordinate::setY(int y) {
    this->y = y;
}
void Coordinate::set(Coordinate f) {
    this->x = f.x;
    this->y = f.y;
}

Rectangle::Rectangle() {
    this->first = Coordinate();
    this->second = Coordinate();
}
Rectangle::Rectangle(Coordinate f, Coordinate s) {
    this->first = Coordinate();
    this->second = Coordinate();
    first.set(f);
    second.set(s);
}
Rectangle::Rectangle(int fX, int fY, int sX, int sY) {
    this->first = Coordinate(fX, fY);
```

```
    this->second = Coordinate(sX, sY);
}
/*
int Rectangle::max(int a, int b) {
    return a > b ? a : b;
}

int Rectangle::min(int a, int b) {
    return a > b ? b : a;
}
*/

Coordinate Rectangle::getFirstCoordinate() {
    return this->first;
}
Coordinate Rectangle::getSecondCoordinate() {
    return this->second;
}

int Rectangle::square() {
    return abs((first.getX() - second.getX())*(first.getY() - second.getY()));
}
int Rectangle::perimetr() {
    return 2 * (abs(first.getX() - second.getX()) + abs(first.getY() - second.getY()));
}

Rectangle Rectangle::intersection(Rectangle rec) {
    int left = min(first.getX(), rec.first.getX());
    int right = max(second.getX(), rec.second.getX());
    int top = max(second.getY(), rec.second.getY());
    int bottom = min(first.getX(), rec.first.getY());

    Coordinate f;
    f.setX(left);
    f.setY(bottom);
    Coordinate s;
    s.setX(right);
    s.setY(top);
    Rectangle result{ f,s };
    return result;

}
Rectangle Rectangle::composition(Rectangle rec) {
    Rectangle result{};
    int left = max(this->first.getX(), rec.first.getX());
```

```
    int top = min(this->second.getY(), rec.second.getY());
    int right = min(this->second.getX(), rec.second.getX());
    int bottom = max(this->first.getY(), rec.first.getY());

    int width = right - left;
    int height = top - bottom;

    if (!((width < 0) || (height < 0))) {
        Coordinate f;
        f.setX(left);
        f.setY(bottom);
        Coordinate s;
        s.setX(right);
        s.setY(top);
        result.changeRectangle(f, s);
    }

    return result;
}

void Rectangle::shift(int onX, int onY) {
    first.setX(first.getX() + onX);
    second.setY(first.getY() + onY);
}
void Rectangle::changeRectangle(Coordinate f, Coordinate s) {
    first.set(f);
    second.set(s);
}
bool Rectangle::compareSquare(Rectangle rec) {
    return this->square() == rec.square();
}
bool Rectangle::comparePerimetr(Rectangle rec) {
    return this->perimetr() == rec.perimetr();
}
```

CMakeLists.txt

```
cmake_minimum_required(VERSION 2.8)  # Проверка версии CMake.
                                     # Если версия
установленой программы
                                     # старее указаной,
произайдёт аварийный выход.

project(lab1)                        # Название проекта
```

```
set(SOURCE_EXE main_io.cpp)                # Установка переменной со списком исходников для исполняемого файла

set(SOURCE_LIB rectangle.cpp)                    # Тоже самое, но для библиотеки

add_library(rectangle STATIC ${SOURCE_LIB})        # Создание статической библиотеки с именем foo

add_executable(main ${SOURCE_EXE})   # Создает исполняемый файл с именем main

target_link_libraries(main rectangle)
```

Test01.txt

```
2 2 3 4
1 2 3 4
```

Test02.txt

```
1 2 3 4
1 2 3 4
```

Test03.txt

```
1 1 1 1
1 1 1 1
```

## 4. Результаты выполнения тестов

Result_test01.txt

```
First Rectangle:
        First Coordinate: X: 2 Y: 2
        Second Coordinate: X: 3 Y: 4
        Square: 2 Perimetr : 6
Second Rectangle:
        First Coordinate: X: 1 Y: 2
        Second Coordinate: X: 3 Y: 4
        Square: 4 Perimetr : 8
Compare on Square: 0 Compare on Perimetr: 0
Intersection :
```

First Coordinate: X: 1 Y: 2
Second Coordinate: X: 3 Y: 4
Square: 4 Perimetr : 8
Composition :
First Coordinate: X: 2 Y: 2
Second Coordinate: X: 3 Y: 4
Square: 2 Perimetr : 6

Result_test02.txt

First Rectangle:
First Coordinate: X: 1 Y: 2
Second Coordinate: X: 3 Y: 4
Square: 4 Perimetr : 8
Second Rectangle:
First Coordinate: X: 1 Y: 2
Second Coordinate: X: 3 Y: 4
Square: 4 Perimetr : 8
Compare on Square: 1 Compare on Perimetr: 1
Intersection :
First Coordinate: X: 1 Y: 1
Second Coordinate: X: 3 Y: 4
Square: 6 Perimetr : 10
Composition :
First Coordinate: X: 1 Y: 2
Second Coordinate: X: 3 Y: 4
Square: 4 Perimetr : 8

Result_test03.txt

First Rectangle:
First Coordinate: X: 1 Y: 1
Second Coordinate: X: 1 Y: 1
Square: 0 Perimetr : 0
Second Rectangle:
First Coordinate: X: 1 Y: 1
Second Coordinate: X: 1 Y: 1
Square: 0 Perimetr : 0
Compare on Square: 1 Compare on Perimetr: 1
Intersection :
First Coordinate: X: 1 Y: 1
Second Coordinate: X: 1 Y: 1
Square: 0 Perimetr : 0
Composition :
First Coordinate: X: 1 Y: 1

Second Coordinate: X: 1 Y: 1
Square: 0 Perimetr : 0

## 5. Объяснение результатов работы программы

Программа просит на вход файл с координатами прямоугольников, далее происходят вывод координат, расчет периметра, пересечения и объединения прямоугольников. Результат записывает в файл с добавлением подстроки result.

## 6. Вывод

Изучили основы объектно-ориентированного программирования, методы, классы, написал простой класс Rectangle, который реализует прямоугольник, который задается с помощью подкласса Coordinate, которой состоит из двух целочисленных переменных X и Y, а также set и get методов. В классе Rectangle определены get и set методы, метод нахождения площади, периметр, объединения прямоугольников и их пересечение, а также перемещение прямоугольника.