**Московский авиационный институт**

**(Национальный исследовательский университет)**

Факультет: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Объектно-ориентированное программирование»

**Лабораторная работа № 1**

Тема: Простые классы на языке С++

Студент: Терво Александр Александрович

Группа: 80-207

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2020

1. Постановка задачи

**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.

1. Описание программы

У класса Rectangle создадим четыре переменных для хранения координат левой нижней и правой верхней точек. Реализуем конструктор, деструктор и метоты, описанные в задании. Для удобства чтения кода, вынесем реализацию методов в отдельный файл.

На вход программе подаются координаты первого прямоугольника, величины, на которые требуется сдвинуть его и изменить его размеры, координары второго прямоугольника. Программа выводит на экран площадь и периметр первого прямоугольника, координаты его точек после перемещения и изменения размеров, площади первого и второго прямоугольников, результат их сравнения, аналогично для периметров, результат пересечения и объединения прямоугольников.

1. Набор тестов

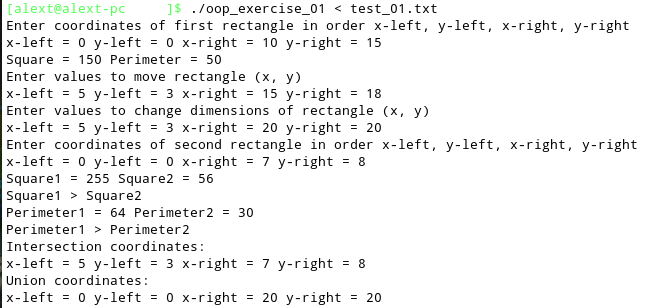
Тест №1

0 0 10 15

5 3

5 2

0 0 7 8

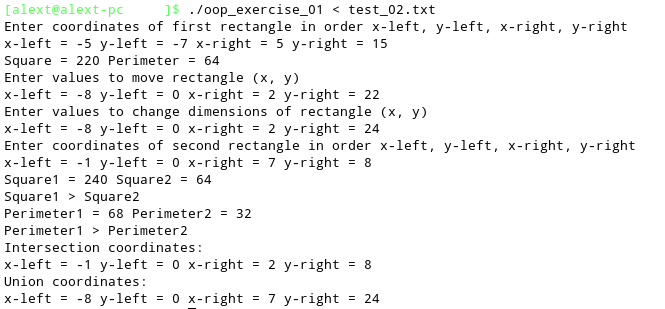
Тест №2

-5 -7 5 15

-3 7

0 2

-1 0 7 8

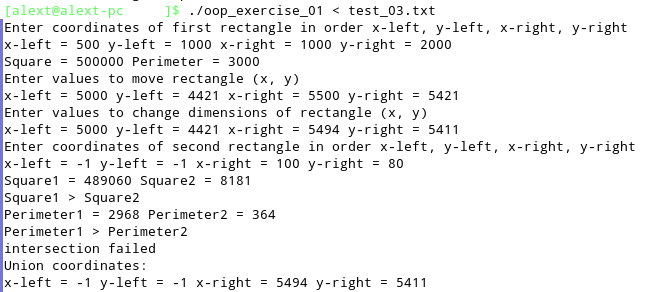
Тест№3

500 6000 1000 10743

4500 3421

-6 -10

-1 -1 100 80



1. Листинг программы

main.cpp:

#include <iostream>

#include "rectangle.hpp"

int main() {

std::cout << "Enter coordinates of first rectangle in order x-left, y-left, x-right, y-right\n";

double xl, xr, yl, yr;

std::cin >> xl >> yl >> xr >> yr;

std::cout << "x-left = " << xl << " y-left = " << yl << " x-right = " << xr << " y-right = " << yr << std::endl;

Rectangle rec = Rectangle(xl, yl, xr, yr);

std::cout << "Square = " << rec.getSquare() << " Perimeter = " << rec.getPerimeter() << std::endl;

std::cout << "Enter values to move rectangle (x, y)\n";

double mvx, mvy;

std::cin >> mvx >> mvy;

rec.move(mvx, mvy);

std::cout << "x-left = " << rec.xl << " y-left = " << rec.yl;

std::cout << " x-right = " << rec.xr << " y-right = " << rec.yr << std::endl;

std::cout << "Enter values to change dimensions of rectangle (x, y)\n";

std::cin >> mvx >> mvy;

rec.changeDimensions(mvx, mvy);

std::cout << "x-left = " << rec.xl << " y-left = " << rec.yl;

std::cout << " x-right = " << rec.xr << " y-right = " << rec.yr << std::endl;

std::cout << "Enter coordinates of second rectangle in order x-left, y-left, x-right, y-right\n";

std::cin >> xl >> yl >> xr >> yr;

std::cout << "x-left = " << xl << " y-left = " << yl << " x-right = " << xr << " y-right = " << yr << std::endl;

Rectangle rec2 = Rectangle(xl, yl, xr, yr);

std::cout << "Square1 = " << rec.getSquare() << " Square2 = " << rec2.getSquare() << std::endl;

std::cout << ((rec.isSquareLarger(rec2)) ? "Square1 > Square2\n" : "Square2 >= Square1\n");

std::cout << "Perimeter1 = " << rec.getPerimeter() << " Perimeter2 = " << rec2.getPerimeter() << std::endl;

std::cout << ((rec.isPerimeterLarger(rec2)) ? "Perimeter1 > Perimeter2\n" : "Perimeter2 >= Perimeter1\n");

bool code;

Rectangle rec3 = rec.intersection(rec2, code);

if (code) {

std::cout << "Intersection coordinates:\n";

std::cout << "x-left = " << rec3.xl << " y-left = " << rec3.yl;

std::cout << " x-right = " << rec3.xr << " y-right = " << rec3.yr << std::endl;

} else {

std::cout << "intersection failed\n";

}

rec3 = rec.uni(rec2);

std::cout << "Union coordinates:\n";

std::cout << "x-left = " << rec3.xl << " y-left = " << rec3.yl;

std::cout << " x-right = " << rec3.xr << " y-right = " << rec3.yr << std::endl;

return 0;

}

rectangle.hpp:

#ifndef RECTANGLE\_HPP

#define RECTANGLE\_HPP

#include <algorithm>

class Rectangle {

private:

bool is\_point\_on(double x, double y, Rectangle &rec);

public:

double xl, yl, xr, yr;

Rectangle(double Xl, double Yl, double Xr, double Yr) : xl(Xl), yl(Yl), xr(Xr), yr(Yr) {};

~Rectangle() {};

double getSquare();

double getPerimeter();

void move(double x, double y);

void changeDimensions(double x, double y);

bool isSquareLarger(Rectangle &rec2);

bool isPerimeterLarger(Rectangle &rec2);

Rectangle intersection(Rectangle rec2, bool &code);

Rectangle uni(Rectangle rec2);

};

#endif

rectangle.cpp:

#include "rectangle.hpp"

double Rectangle::getSquare() {

return (xr - xl) \* (yr - yl);

}

double Rectangle::getPerimeter() {

return 2 \* (xr - xl + yr - yl);

}

void Rectangle::move(double x, double y) { //на сколько по x и y сдвинуть

xr += x;

xl += x;

yr += y;

yl += y;

}

void Rectangle::changeDimensions(double x, double y) { //увеличить на х по ОХ и на y по OY

xr += x;

yr += y;

}

bool Rectangle::isSquareLarger(Rectangle &rec2) {

double square1 = getSquare();

double square2 = rec2.getSquare();

if (square1 > square2) {

return true;

} else {

return false;

}

}

bool Rectangle::isPerimeterLarger(Rectangle &rec2) {

double perimeter1 = getPerimeter();

double perimeter2 = rec2.getPerimeter();

if (perimeter1 > perimeter2) {

return true;

} else {

return false;

}

}

Rectangle Rectangle::intersection(Rectangle rec2, bool &code) {

code = true;

double int\_x\_left = 0;

double int\_x\_right = 0;

double int\_y\_left = 0;

double int\_y\_right = 0;

bool a, b, c, d;

a = is\_point\_on(xl, yr, rec2); //upper left

b = is\_point\_on(xr, yr, rec2); //upper right

c = is\_point\_on(xl, yl, rec2); //down left

d = is\_point\_on(xr, yl, rec2); //down right

if (a) {

//std::cout << "a is on the rec2\n";

if (b) { //rec2 limits only on bottom

//std::cout << "ab is on the rec2\n";

if (d) { //that means that rec fully placed on rec2

//std::cout << "abcd is on the rec2\n";

int\_x\_left = xl;

int\_y\_left = yl;

int\_x\_right = xr;

int\_y\_right = yr;

} else {

int\_x\_left = xl;

int\_y\_left = rec2.yl;

int\_x\_right = xr;

int\_y\_right = yr;

}

} else

if (c) { //rec2 limits only on right border

//std::cout << "ac is on the rec2\n";

int\_x\_left = xl;

int\_y\_left = yl;

int\_x\_right = rec2.xr;

int\_y\_right = yr;

} else { //that means only a placed on the rec2

//std::cout << "a is on the rec2\n";

int\_x\_left = xl;

int\_y\_left = rec2.yl;

int\_x\_right = rec2.xr;

int\_y\_right = yr;

}

} else

if (b) {

//std::cout << "b is on the rec2\n";

if (c) {

//std::cout << "bc is on the rec2\n";

int\_x\_left = rec2.xl;

int\_y\_left = yl;

int\_x\_right = xr;

int\_y\_right = yr;

} else { //only b

//std::cout << "b is on the rec2\n";

int\_x\_left = rec2.xl;

int\_y\_left = rec2.yl;

int\_x\_right = xr;

int\_y\_right = yr;

}

} else

if (c) {

//std::cout << "c is on the rec2\n";

if (d) {

//std::cout << "cd is on the rec2\n";

int\_x\_left = xl;

int\_y\_left = yl;

int\_x\_right = xr;

int\_y\_right = rec2.yr;

} else { //only c

//std::cout << "c is on the rec2\n";

int\_x\_left = xl;

int\_y\_left = yl;

int\_x\_right = rec2.xr;

int\_y\_right = rec2.yr;

}

} else

if (d) { //only d

//std::cout << "d is on the rec2\n";

int\_x\_left = rec2.xl;

int\_y\_left = yl;

int\_x\_right = xr;

int\_y\_right = rec2.yr;

} else

if (is\_point\_on(rec2.xl, rec2.yr, \*this)) {

int\_x\_left = rec2.xl;

int\_y\_left = rec2.yl;

int\_x\_right = rec2.xr;

int\_y\_right = rec2.yr;

} else {

code = false;

}

//std::cout << "a = " << a << " b = " << b << " c = " << c << " d = " << d << " code = " << code << std::endl;

Rectangle rec\_int = Rectangle(int\_x\_left, int\_y\_left, int\_x\_right, int\_y\_right);

//code = true;

return rec\_int;

}

Rectangle Rectangle::uni(Rectangle rec2) {

double uxl, uyl, uxr, uyr;

uxl = std::min(xl, rec2.xl);

uyl = std::min(yl, rec2.yl);

uxr = std::max(xr, rec2.xr);

uyr = std::max(yr, rec2.yr);

Rectangle rec\_un = Rectangle(uxl, uyl, uxr, uyr);

return rec\_un;

}

bool Rectangle::is\_point\_on(double x, double y, Rectangle &rec) {

if ((x >= rec.xl) && (x <= rec.xr) && (y >= rec.yl) && (y <= rec.yr)) {

return true;

} else {

return false;

}

}

5. Выводы

В ходе работы я изучил процесс создания классов, описания их членов и методов на языке

С++, также была освоена автоматическая система сборки Cmake и основы работы с системой контроля версий GIT.