**ОТЧЕТ**

**ЛАБОРАТОРНАЯ РАБОТА №2**

**Вариант №12**

**Выполнил:**

**Студент гр. ДИНРБ-21**

**Яхьяев Данис Дамирович**

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.

**Файл заголовка «Rectangle.h»**

#ifndef RECTANGLE\_H

#define RECTANGLE\_H

#include <iostream>

using namespace std;

class Rectangle {

private:

double x1, y1; // координаты левого нижнего угла

double x2, y2; // координаты правого верхнего угла

public:

// Конструкторы

Rectangle();

Rectangle(double x1, double y1, double x2, double y2);

// Методы для получения параметров прямоугольника

double getX1() const noexcept;

double getY1() const noexcept;

double getX2() const noexcept;

double getY2() const noexcept;

// Методы для вычисления площади и периметра

double calculateArea() const noexcept;

double calculatePerimeter() const noexcept;

// Методы для перемещения, изменения размеров и сравнения

void move(double dx, double dy);

void resize(double dw, double dh);

bool operator==(const Rectangle& other) const noexcept;

bool operator<(const Rectangle& other) const noexcept;

// Метод для получения прямоугольника пересечения

Rectangle getIntersection(const Rectangle& other) const;

// Метод для объединения двух прямоугольников

Rectangle getUnion(const Rectangle& other) const;

// Перегрузка операций ввода/вывода

friend istream& operator>>(istream& in, Rectangle& rectangle);

friend ostream& operator<<(ostream& out, const Rectangle& rectangle);

// Метод преобразования в строку

string toString() const;

};

#endif // RECTANGLE\_H

**Файл реализации «Rectangle.cpp»**

#include "Rectangle.h"

#include <string>

using namespace std;

// Конструкторы

Rectangle::Rectangle() : x1(0), y1(0), x2(0), y2(0) {}

Rectangle::Rectangle(double x1, double y1, double x2, double y2)

: x1(x1), y1(y1), x2(x2), y2(y2) {}

// Методы для получения параметров прямоугольника

double Rectangle::getX1() const noexcept {

return x1;

}

double Rectangle::getY1() const noexcept {

return y1;

}

double Rectangle::getX2() const noexcept {

return x2;

}

double Rectangle::getY2() const noexcept {

return y2;

}

// Методы для вычисления площади и периметра

double Rectangle::calculateArea() const noexcept {

return (x2 - x1) \* (y2 - y1);

}

double Rectangle::calculatePerimeter() const noexcept {

return 2 \* ((x2 - x1) + (y2 - y1));

}

// Методы для перемещения, изменения размеров и сравнения

void Rectangle::move(double dx, double dy) {

x1 += dx;

y1 += dy;

x2 += dx;

y2 += dy;

}

void Rectangle::resize(double dw, double dh) {

x2 += dw;

y2 += dh;

}

bool Rectangle::operator==(const Rectangle& other) const noexcept {

return calculateArea() == other.calculateArea();

}

bool Rectangle::operator<(const Rectangle& other) const noexcept {

return calculatePerimeter() < other.calculatePerimeter();

}

// Метод для получения прямоугольника пересечения

Rectangle Rectangle::getIntersection(const Rectangle& other) const {

double intersectX1 = max(x1, other.x1);

double intersectY1 = max(y1, other.y1);

double intersectX2 = min(x2, other.x2);

double intersectY2 = min(y2, other.y2);

if (intersectX1 < intersectX2 && intersectY1 < intersectY2) {

return Rectangle(intersectX1, intersectY1, intersectX2, intersectY2);

}

else {

// Возвращаем пустой прямоугольник, если пересечения нет

return Rectangle();

}

}

// Метод для объединения двух прямоугольников

Rectangle Rectangle::getUnion(const Rectangle& other) const {

double unionX1 = min(x1, other.x1);

double unionY1 = min(y1, other.y1);

double unionX2 = max(x2, other.x2);

double unionY2 = max(y2, other.y2);

return Rectangle(unionX1, unionY1, unionX2, unionY2);

}

// Перегрузка операций ввода/вывода

istream& operator>>(istream& in, Rectangle& rectangle) {

in >> rectangle.x1 >> rectangle.y1 >> rectangle.x2 >> rectangle.y2;

return in;

}

ostream& operator<<(ostream& out, const Rectangle& rectangle) {

out << "Rectangle: (" << rectangle.x1 << ", " << rectangle.y1 << ") - ("

<< rectangle.x2 << ", " << rectangle.y2 << ")";

return out;

}

// Метод преобразования в строку

string Rectangle::toString() const {

return "(" + to\_string(x1) + ", " + to\_string(y1) + ") - ("

+ to\_string(x2) + ", " + to\_string(y2) + ")";

}

**Файл главной функции «main.cpp»**

#include "Rectangle.h"

#include <iostream>

#include <cassert>

using namespace std;

int main() {

setlocale(LC\_ALL, "RU");

try {

// Создание объектов класса Rectangle

Rectangle rect1(0, 0, 5, 5);

Rectangle rect2(2, 2, 8, 8);

// Проверки с использованием assert

assert(rect1.calculateArea() == 25.0);

assert(rect1.calculatePerimeter() == 20.0);

assert(rect2.calculateArea() == 36.0);

assert(rect2.calculatePerimeter() == 24.0);

// Вывод параметров прямоугольников

cout << "Прямоугольник 1: " << rect1 << endl;

cout << "Прямоугольник 2: " << rect2 << endl;

// Вычисление и вывод площади и периметра

cout << "Площадь прямоугольника 1: " << rect1.calculateArea() << endl;

cout << "Периметр прямоугольника 1: " << rect1.calculatePerimeter() << endl;

cout << "Площадь прямоугольника 2: " << rect2.calculateArea() << endl;

cout << "Периметр прямоугольника 2: " << rect2.calculatePerimeter() << endl;

// Перемещение и изменение размеров прямоугольника

rect1.move(2, 3);

rect2.resize(3, 2);

// Проверки с использованием assert после изменений

assert(rect1.getX1() == 2.0);

assert(rect2.getY2() == 10.0);

// Вывод измененных параметров

cout << "Перемещенный прямоугольник 1: " << rect1 << endl;

cout << "Измененный размер прямоугольника 2: " << rect2 << endl;

// Сравнение прямоугольников

if (rect1 == rect2) {

cout << "Прямоугольники равны по площади." << endl;

}

else if (rect1 < rect2) {

cout << "У прямоугольника 1 меньший периметр, чем у прямоугольника 2." << endl;

}

else {

cout << "У прямоугольника 1 больший периметр, чем у прямоугольника 2." << endl;

}

// Получение прямоугольника пересечения

Rectangle intersection = rect1.getIntersection(rect2);

cout << "Прямоугольник пересечения: " << intersection << endl;

// Получение объединения двух прямоугольников

Rectangle unionRect = rect1.getUnion(rect2);

cout << "Прямоугольник объединения: " << unionRect << endl;

}

catch (const exception& e) {

cerr << "Исключение: " << e.what() << endl;

}

return 0;

}