***Отчет по 3 вариант Класс «Прямоугольник, со сторонами, параллельными координатным осям»***

*Член-данные класса*

1) координаты точки A;

2) длины сторон a (горизонтальной) и b (вертикальной).

*Методы класса*

1) конструкторы: по умолчанию, с аргументами (или с аргументами по умолчанию);

2) ввод/вывод прямоугольника;

3) нахождение площади;

4) нахождение радиуса описанной окружности;

5) проверка, является ли прямоугольник квадратом;

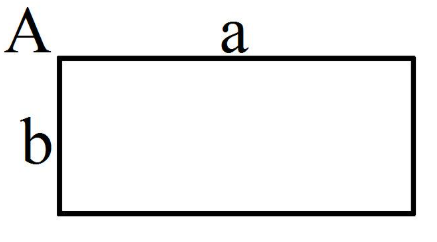
6) нахождение произведения прямоугольника на число (координаты точки A не изменяются, длины сторон умножаются на число);

7) проверка двух прямоугольников на равенство (прямоугольники равны, если один получается из другого параллельным переносом);

8) проверка, расположен ли прямоугольник полностью в первой четверти;

9) проверка пересечения прямоугольников;

10\*) проверка, пересекает ли заданная прямая прямоугольник.



**Код программы:**

using System;

public class Point

{

public double X { get; set; }

public double Y { get; set; }

public Point(double x = 0, double y = 0)

{

X = x;

Y = y;

}

public override string ToString()

{

return $"({X:F2}, {Y:F2})";

}

}

public class Rectangle

{

private Point pointA; // Координаты левого нижнего угла

private double sideA; // Длина горизонтальной стороны

private double sideB; // Длина вертикальной стороны

// 1) Конструкторы

// Конструктор по умолчанию - создает единичный квадрат в начале координат

public Rectangle()

{

pointA = new Point(0, 0);

sideA = 1;

sideB = 1;

}

// Конструктор с аргументами

public Rectangle(double x, double y, double a, double b)

{

if (a <= 0 || b <= 0)

throw new ArgumentException("стороны должны быть положительными");

pointA = new Point(x, y);

sideA = a;

sideB = b;

}

// Конструктор с точкой

public Rectangle(Point point, double a, double b)

{

if (a <= 0 || b <= 0)

throw new ArgumentException("стороны должны быть положительными");

pointA = new Point(point.X, point.Y);

sideA = a;

sideB = b;

}

// 2) Методы ввода/вывода

public void Input()

{

Console.WriteLine("--- VVOD ---");

Console.Write("введите X точки A (левый нижний угол): ");

double x = Convert.ToDouble(Console.ReadLine());

Console.Write("введите Y точки A (левый нижний угол): ");

double y = Convert.ToDouble(Console.ReadLine());

Console.Write("введите длину горизонтальной стороны a: ");

double a = Convert.ToDouble(Console.ReadLine());

Console.Write("введите длину вертикальной стороны b: ");

double b = Convert.ToDouble(Console.ReadLine());

if (a <= 0 || b <= 0)

{

Console.WriteLine("стороны должны быть положительными");

return;

}

pointA = new Point(x, y);

sideA = a;

sideB = b;

}

public void Output()

{

Console.WriteLine("--- Прямоугольник ---");

Console.WriteLine($"точка A (левый нижний угол): {pointA}");

Console.WriteLine($"горизонтальная сторона: {sideA:F2}");

Console.WriteLine($"вертикальная сторона: {sideB:F2}");

Console.WriteLine($"Координаты углов:");

Console.WriteLine($"\t A (лев.низ): {pointA}");

Console.WriteLine($"\t B (прав.низ): ({pointA.X + sideA:F2}, {pointA.Y:F2})");

Console.WriteLine($"\t C (прав.верх): ({pointA.X + sideA:F2}, {pointA.Y + sideB:F2})");

Console.WriteLine($"\t D (лев.верх): ({pointA.X:F2}, {pointA.Y + sideB:F2})");

}

// 3) Нахождение площади

public double GetArea()

{

return sideA \* sideB;

}

// 4) Нахождение радиуса описанной окружности

// для прямоугольника радиус описанной окружности = половина диагонали

public double GetCircumscribedRadius()

{

double diagonal = Math.Sqrt(sideA \* sideA + sideB \* sideB);

return diagonal / 2.0;

}

// 5) Проверка, является ли прямоугольник квадратом

public bool IsSquare()

{

// используем небольшую погрешность для сравнения вещественных чисел

return Math.Abs(sideA - sideB) < 1e-10;

}

// 6) Произведение прямоугольника на число

public Rectangle MultiplyByNumber(double multiplier)

{

if (multiplier <= 0)

throw new ArgumentException("множитель должен быть положительным");

// Точка A не изменяется, стороны умножаются на число

return new Rectangle(pointA.X, pointA.Y, sideA \* multiplier, sideB \* multiplier);

}

// 7) Проверка равенства прямоугольников (с учетом параллельного переноса)

public bool Equals(Rectangle other)

{

if (other == null) return false;

// прямоугольники равны, если имеют одинаковые размеры

// координаты могут отличаться из-за параллельного переноса

return Math.Abs(sideA - other.sideA) < 1e-10 &&

Math.Abs(sideB - other.sideB) < 1e-10;

}

// 8) Проверка, расположен ли прямоугольник полностью в первой четверти

public bool IsInFirstQuadrant()

{

// первая четверть: x >= 0, y >= 0

// проверяем все четыре угла

return pointA.X >= 0 && pointA.Y >= 0 && // левый нижний

(pointA.X + sideA) >= 0 && pointA.Y >= 0 && // правый нижний

(pointA.X + sideA) >= 0 && (pointA.Y + sideB) >= 0 && // правый верхний

pointA.X >= 0 && (pointA.Y + sideB) >= 0; // левый верхний

}

// 9) Проверка пересечения с другим прямоугольником

public bool IntersectsWith(Rectangle other)

{

if (other == null) return false;

// Границы текущего прямоугольника

double left1 = pointA.X;

double right1 = pointA.X + sideA;

double bottom1 = pointA.Y;

double top1 = pointA.Y + sideB;

// Границы другого прямоугольника

double left2 = other.pointA.X;

double right2 = other.pointA.X + other.sideA;

double bottom2 = other.pointA.Y;

double top2 = other.pointA.Y + other.sideB;

// не пересекаются, если один полностью слева, справа, сверху или снизу от другого

return !(right1 <= left2 || right2 <= left1 || top1 <= bottom2 || top2 <= bottom1);

}

// 10\*) Проверка пересечения с прямой

// прямая задается уравнением ax + by + c = 0

public bool IntersectsWithLine(double a, double b, double c)

{

// вычисляем значения функции прямой в четырех углах прямоугольника

double[] corners = new double[4];

corners[0] = a \* pointA.X + b \* pointA.Y + c; // левый нижний

corners[1] = a \* (pointA.X + sideA) + b \* pointA.Y + c; // правый нижний

corners[2] = a \* (pointA.X + sideA) + b \* (pointA.Y + sideB) + c; // правый верхний

corners[3] = a \* pointA.X + b \* (pointA.Y + sideB) + c; // левый верхний

// если есть углы с разными знаками, то прямая пересекает прямоугольник

bool hasPositive = false;

bool hasNegative = false;

foreach (double value in corners)

{

if (Math.Abs(value) < 1e-10) // Угол лежит на прямой

return true;

if (value > 0) hasPositive = true;

if (value < 0) hasNegative = true;

}

return hasPositive && hasNegative;

}

// Дополнительные служебные методы

public Point GetPointA() => new Point(pointA.X, pointA.Y);

public double GetSideA() => sideA;

public double GetSideB() => sideB;

}

// Класс для демонстрации работы

class Program

{

static void Main(string[] args)

{

Console.OutputEncoding = System.Text.Encoding.UTF8;

try

{

Rectangle rect1 = new Rectangle(); // по умолчанию

Console.WriteLine("прямоугольник по умолчанию:");

rect1.Output();

Console.WriteLine("\n" + "-".PadRight(50, '-'));

// Создание прямоугольника с вводом данных

Rectangle rect2 = new Rectangle();

rect2.Input();

rect2.Output();

Console.WriteLine("\n--- тестирование методов ---");

Console.WriteLine($"Площадь: {rect2.GetArea():F2}");

Console.WriteLine($"Радиус описанной окружности: {rect2.GetCircumscribedRadius():F2}");

Console.WriteLine($"Является квадратом: {(rect2.IsSquare() ? "Да" : "Нет")}");

Console.Write("Введите число для умножения размеров: ");

double multiplier = Convert.ToDouble(Console.ReadLine());

Rectangle multipliedRect = rect2.MultiplyByNumber(multiplier);

Console.WriteLine("Результат умножения:");

multipliedRect.Output();

Console.WriteLine($"расположен в первой четверти: {(rect2.IsInFirstQuadrant() ? "Да" : "Нет")}");

Console.WriteLine("\nвведите данные второго прямоугольника для проверки пересечения:");

Rectangle rect3 = new Rectangle();

rect3.Input();

Console.WriteLine($"Прямоугольники пересекаются: {(rect2.IntersectsWith(rect3) ? "Да" : "Нет")}");

Console.WriteLine($"Прямоугольники равны: {(rect2.Equals(rect3) ? "Да" : "Нет")}");

Console.WriteLine("\nПроверка пересечения с прямой ax + by + c = 0:");

Console.Write("Введите коэффициент a: ");

double a = Convert.ToDouble(Console.ReadLine());

Console.Write("Введите коэффициент b: ");

double b = Convert.ToDouble(Console.ReadLine());

Console.Write("Введите коэффициент c: ");

double c = Convert.ToDouble(Console.ReadLine());

Console.WriteLine($"Прямая пересекает прямоугольник: {(rect2.IntersectsWithLine(a, b, c) ? "Да" : "Нет")}");

}

catch (Exception ex)

{

Console.WriteLine($"ошибка: {ex.Message}");

}

}

}

**Тестирование:**

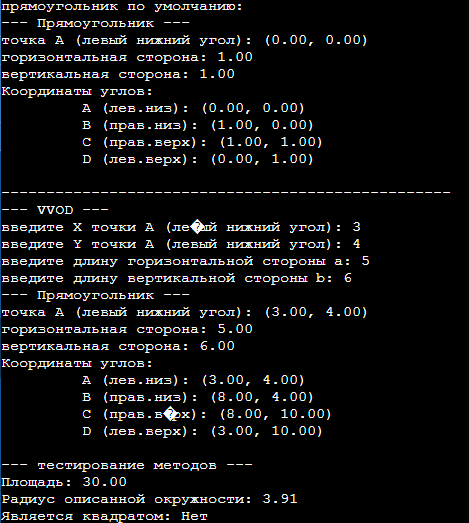
1) конструкторы: по умолчанию, с аргументами (или с аргументами по умолчанию);

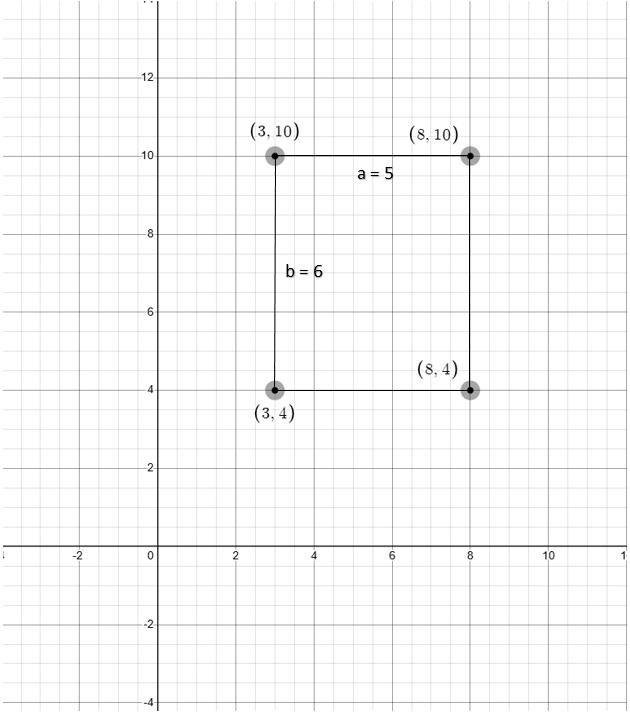
2) ввод/вывод прямоугольника;

3) нахождение площади;

4) нахождение радиуса описанной окружности;

5) проверка, является ли прямоугольник квадратом;





C

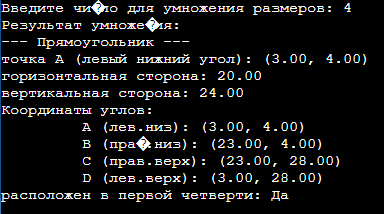
D

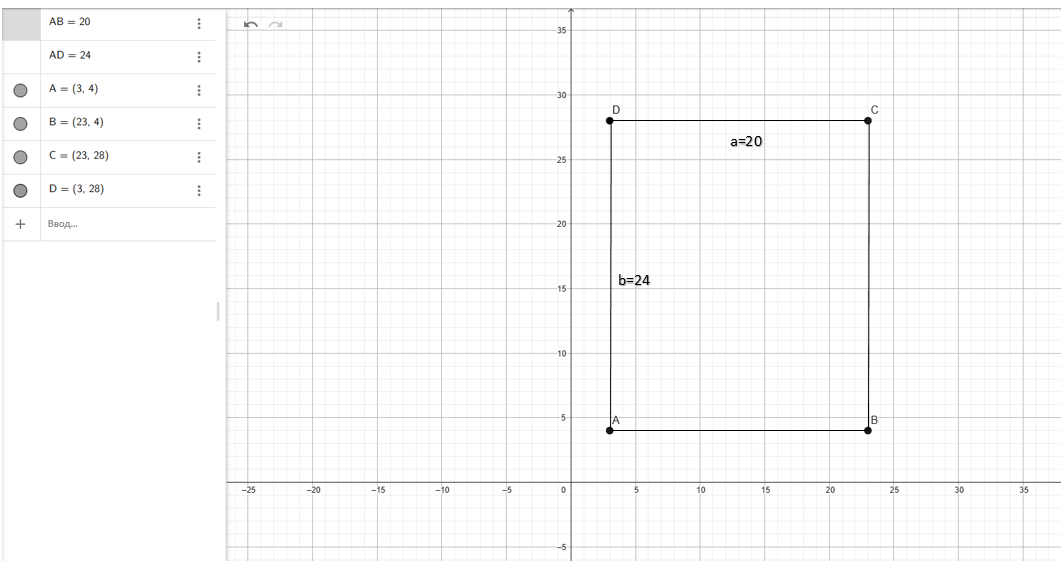
B

A

6) нахождение произведения прямоугольника на число (координаты точки A не изменяются, длины сторон умножаются на число);

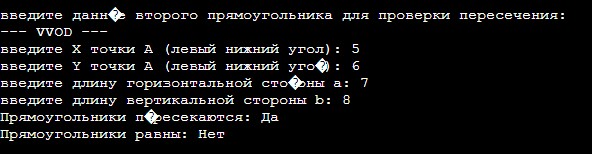
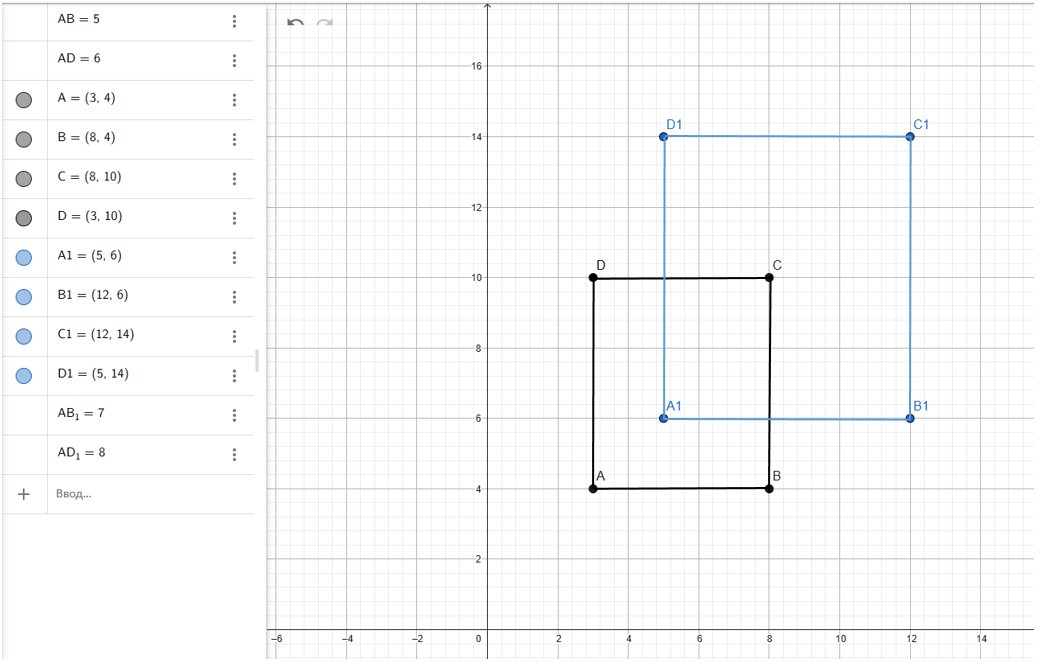
8) проверка, расположен ли прямоугольник полностью в первой четверти;

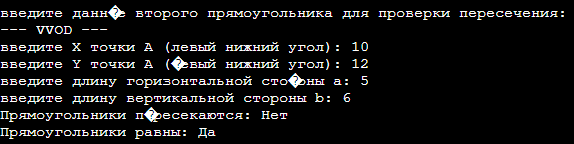


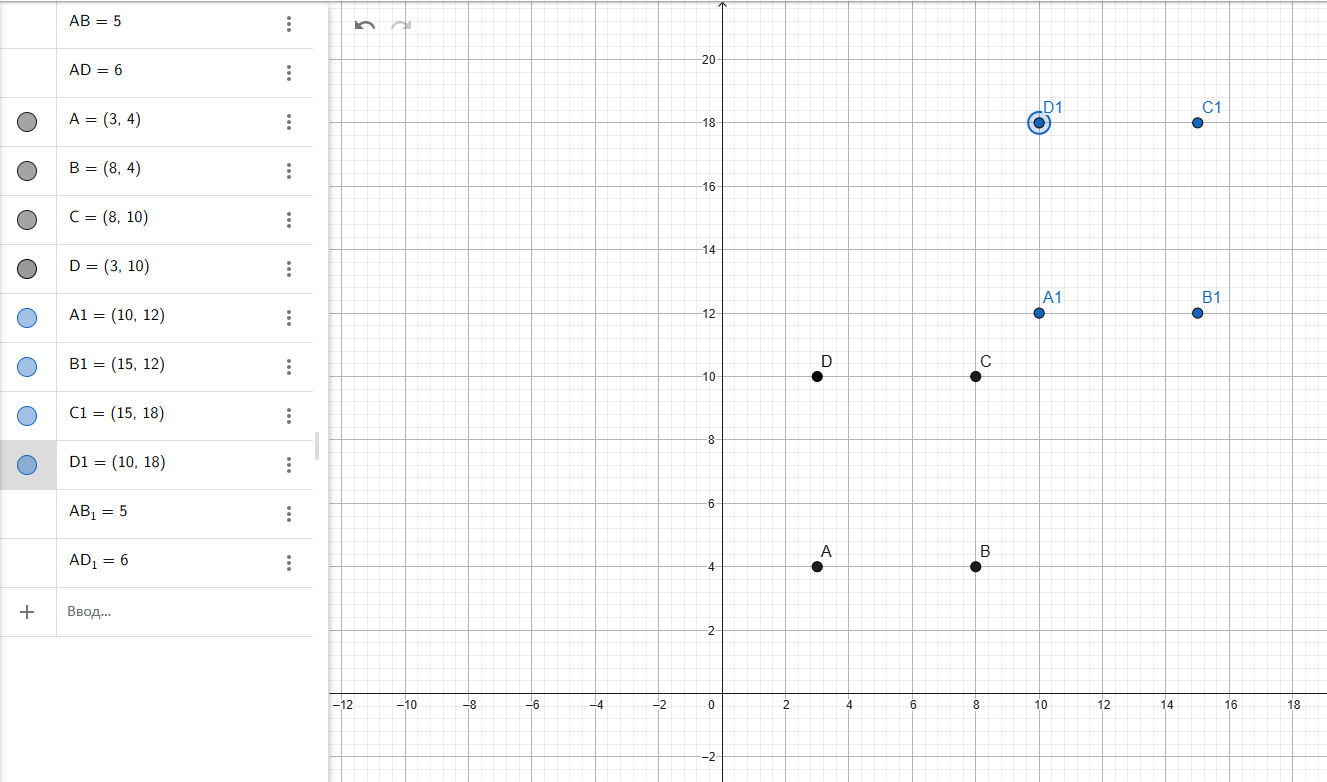


7) проверка двух прямоугольников на равенство (прямоугольники равны, если один получается из другого параллельным переносом);

9) проверка пересечения прямоугольников;







10\*) проверка, пересекает ли заданная прямая прямоугольник.

