Министерство образования и молодежной политики Свердловской области

ГАПОУ СО «Екатеринбургский колледж транспортного строительства»

Отчёт по программе «**Практическое занятие №7.3**»

Выполнила: Яковлева А.Н.

Группа: ПР-21

Преподаватель: Мирошниченко Г.В

2025

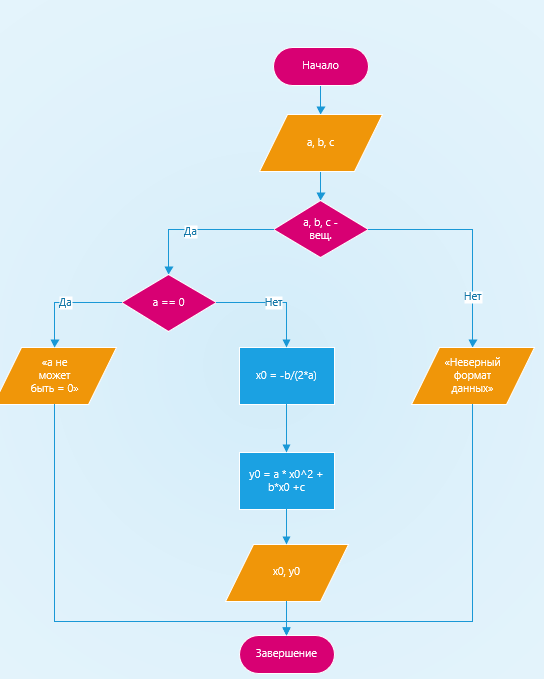
**Задание 8-1:** Определить координаты вершины параболы y = ax2+bx+c (a≠0). Коэффициенты a, b, c ввести.

**Входные и выходные данные**

a, b, c - коэффициенты параболы, вещественный;

x0, y0 - координаты вершины параболы, вещественный.

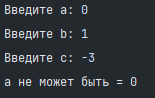
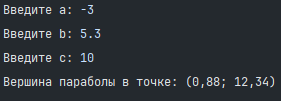
**Блок-схема**

****

**Листинг программы**

fun main() {  
 try {  
 *print*("Введите a: ")  
 val a = *readln*().*toDouble*()  
 *print*("Введите b: ")  
 val b = *readln*().*toDouble*()  
 *print*("Введите c: ")  
 val c = *readln*().*toDouble*()  
 when (a){  
 0.0 -> *println*("а не может быть = 0")  
 else -> {  
 val x0 = -b/(2\*a)  
 val y0 = a \* Math.pow(x0, 2.0) + b \* x0 + c  
 *println*("Вершина параболы в точке: (${String.*format*("%.2f", x0)}; ${String.*format*("%.2f", y0)})")  
 }  
 }  
 }  
 catch(e: Exception)  
 {  
 *println*("Неверный формат данных")  
 }  
}

**Тестовые ситуации**

****

**Задание 18-1:** Дана сторона равностороннего треугольника. Найти его площадь и радиусы описанной и вписанной окружностей.

**Входные и выходные данные:**

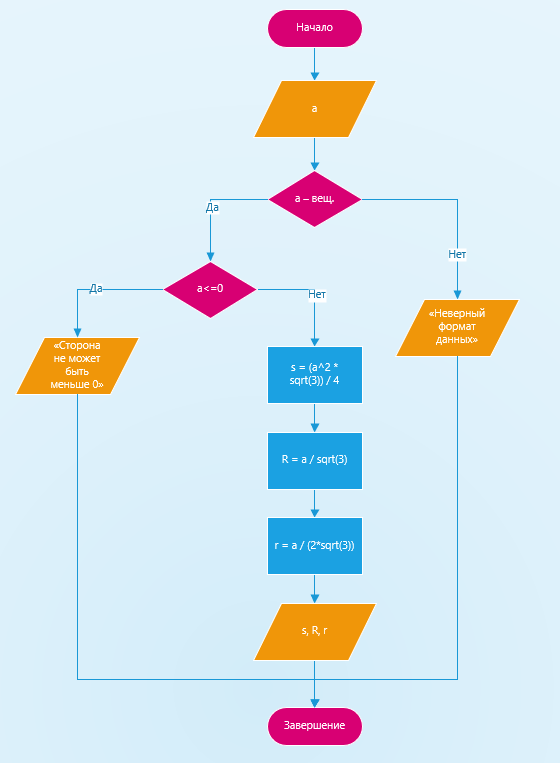
a - сторона треугольника, вещественный;

s - площадь треугольника, вещественный;

r - радиус вписанной окружности, вещественный;

R - радиус описанной окружности, вещественный.

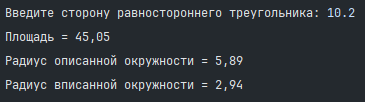
**Блок-схема:**

****

**Листинг программы:**

fun main(){  
 try {  
 *print*("Введите сторону равностороннего треугольника: ")  
 val a = *readln*().*toDouble*()  
 when {  
 a<=0 -> *println*("Сторона не может быть меньше 0")  
 else -> {  
 val s = (Math.pow(a, 2.0) \* Math.sqrt(3.0)) / 4  
 val R = a / Math.sqrt(3.0)  
 val r = a/(2 \* Math.sqrt(3.0))  
 *println*("Площадь = ${String.*format*("%.2f", s)}\nРадиус описанной окружности = ${String.*format*("%.2f", R)}\nРадиус вписанной окружности = ${String.*format*("%.2f", r)}")  
 }  
 }  
 }  
 catch (e: Exception)  
 {  
 *println*("Неверный формат данных")  
 }  
}

**Тестовые ситуации:**

****

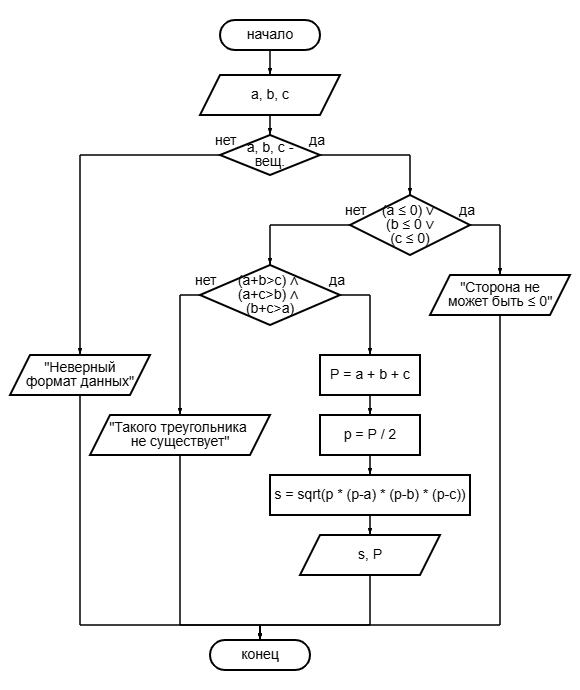
**Задание 8-2:** Даны вещественные числа a, b, c. Если они могут быть длинами сторон, то вычислить периметр и площадь треугольника.

**Входные и выходные данные:**

a, b, c - стороны треугольника, вещественный;

s - площадь треугольника, вещественный;

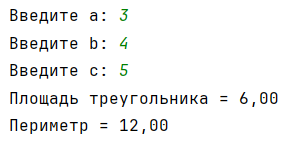
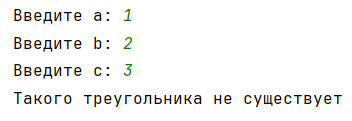
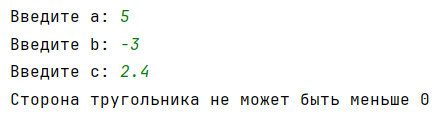
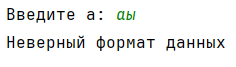
P - периметр треугольника, вещественный.

**Блок-схема:**

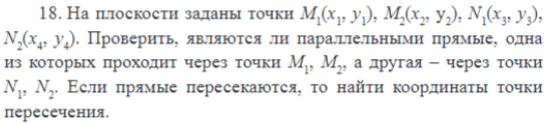
**Листинг программы:**

fun main() {  
 try {  
 *print*("Введите а: ")  
 val a = *readln*().*toDouble*()  
 *print*("Введите b: ")  
 val b = *readln*().*toDouble*()  
 *print*("Введите c: ")  
 val c = *readln*().*toDouble*()  
 when {  
 (a<=0) || (b<=0) || (c<=0) -> *println*("Сторона тругольника не может быть меньше 0")  
 (a+b>c) && (a+c>b) && (b+c>a) -> {  
 val P = a + b + c  
 val p = P / 2  
 val s = Math.sqrt(p \* (p - a) \* (p - b) \* (p - c))  
 *println*("Площадь треугольника = ${String.*format*("%.2f", s)}\nПериметр = ${String.*format*("%.2f", P)}")  
 }  
 else -> *println*("Такого треугольника не существует")  
 }  
 }  
 catch (e: NumberFormatException)  
 {  
 *println*("Неверный формат данных")  
 }  
}

**Тестовые ситуации:**



**Задание 18-2:**



**Входные и выходные данные:**

х1, y1 – координаты точки М1, вещественный;

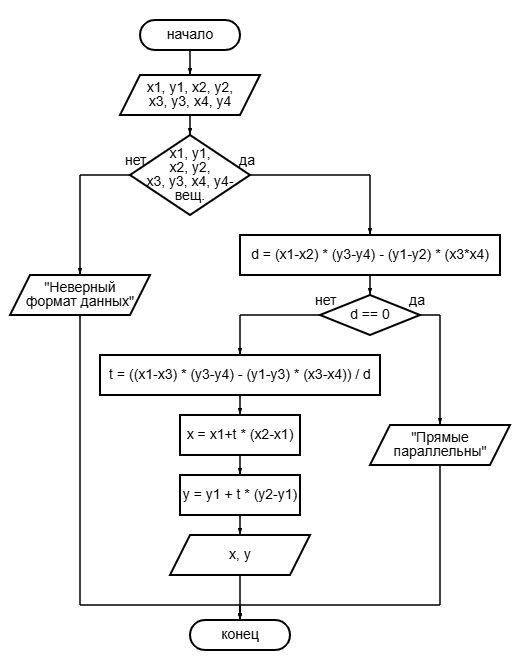
х2, y2 – координаты точки М2, вещественный;

х3, y3 – координаты точки N1, вещественный;

х4, y4 – координаты точки N2, вещественный;

х, y – координаты точки пересечения, вещественный.

**Блок-схема:**

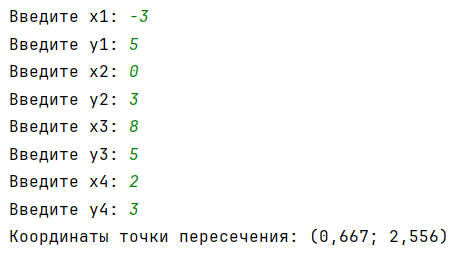
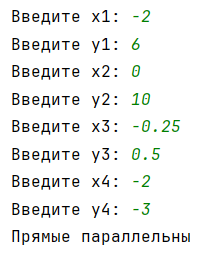


**Листинг программы:**

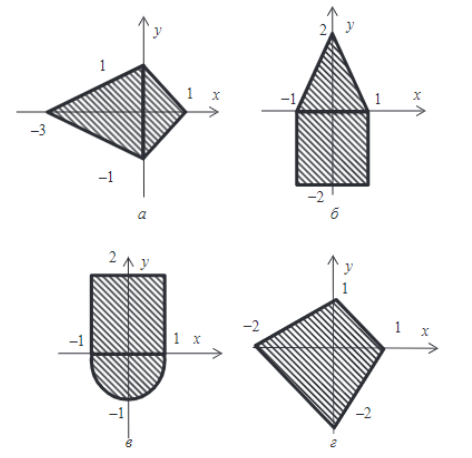
fun main() {  
 try {  
 *//M1*  
 *print*("Введите х1: ")  
 val x1 = *readln*().*toDouble*()  
 *print*("Введите у1: ")  
 val y1 = *readln*().*toDouble*()  
 *//M2*  
 *print*("Введите х2: ")  
 val x2 = *readln*().*toDouble*()  
 *print*("Введите у2: ")  
 val y2 = *readln*().*toDouble*()  
 *//N1*  
 *print*("Введите х3: ")  
 val x3 = *readln*().*toDouble*()  
 *print*("Введите у3: ")  
 val y3 = *readln*().*toDouble*()  
 *//N2*  
 *print*("Введите х4: ")  
 val x4 = *readln*().*toDouble*()  
 *print*("Введите у4: ")  
 val y4 = *readln*().*toDouble*()  
  
 val d = (x1-x2) \* (y3-y4) - (y1-y2) \* (x3-x4)  
 when(d){  
 0.0 -> *println*("Прямые параллельны")  
 else -> {  
 val t = ((x1-x3) \* (y3-y4) - (y1-y3) \* (x3-x4)) / d  
 val x = x1+t \* (x2-x1)  
 val y = y1 + t \* (y2-y1)  
 *println*("Координаты точки пересечения: (${String.*format*("%.3f", x)}; ${String.*format*("%.3f", y)})")  
 }  
 }  
 }  
 catch (e: Exception)  
 {  
 *println*("Неверный формат данных")  
 }  
}

**Тестовые ситуации:**





**Задание 23:** Даны координаты точки на плоскости. Выяснить, принадлежит ли точка закрашенной области в вариантах а-г на рисунке. Ответ выдать в виде строки.



**Входные и выходные данные функции main:**

х, y – координаты точки, вещественный;

Сообщение “Точка (не) принадлежит фигуре”

**Входные и выходные данные функции IsBelongsVM4:**

ax, ay - координаты вершины А, вещественный;

bx, by - координаты вершины B, вещественный;

cx, cy - координаты вершины C, вещественный;

dx, dy - координаты вершины D, вещественный;

x y - координаты точки, введенные пользователем вещественный;

is\_belong - принадлежит ли точка фигуре, логический.

**Входные и выходные данные функции IsBelongsVM5:**

ax, ay - координаты вершины А, вещественный;

bx, by - координаты вершины B, вещественный;

cx, cy - координаты вершины C, вещественный;

dx, dy - координаты вершины D, вещественный;

ex, ey - координаты вершины E, вещественный;

x y - координаты точки, введенные пользователем вещественный;

is\_belong - принадлежит ли точка фигуре, логический.

**Входные и выходные данные функции VectorMultiply:**

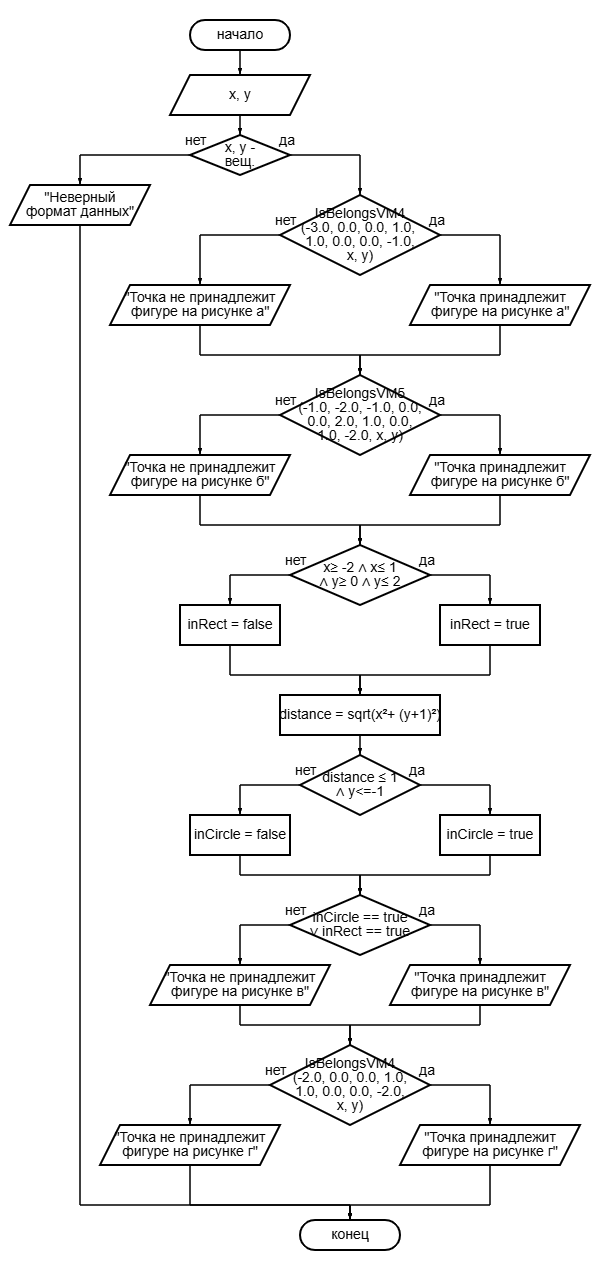
x1, y1 - координаты одной вершины, вещественный;

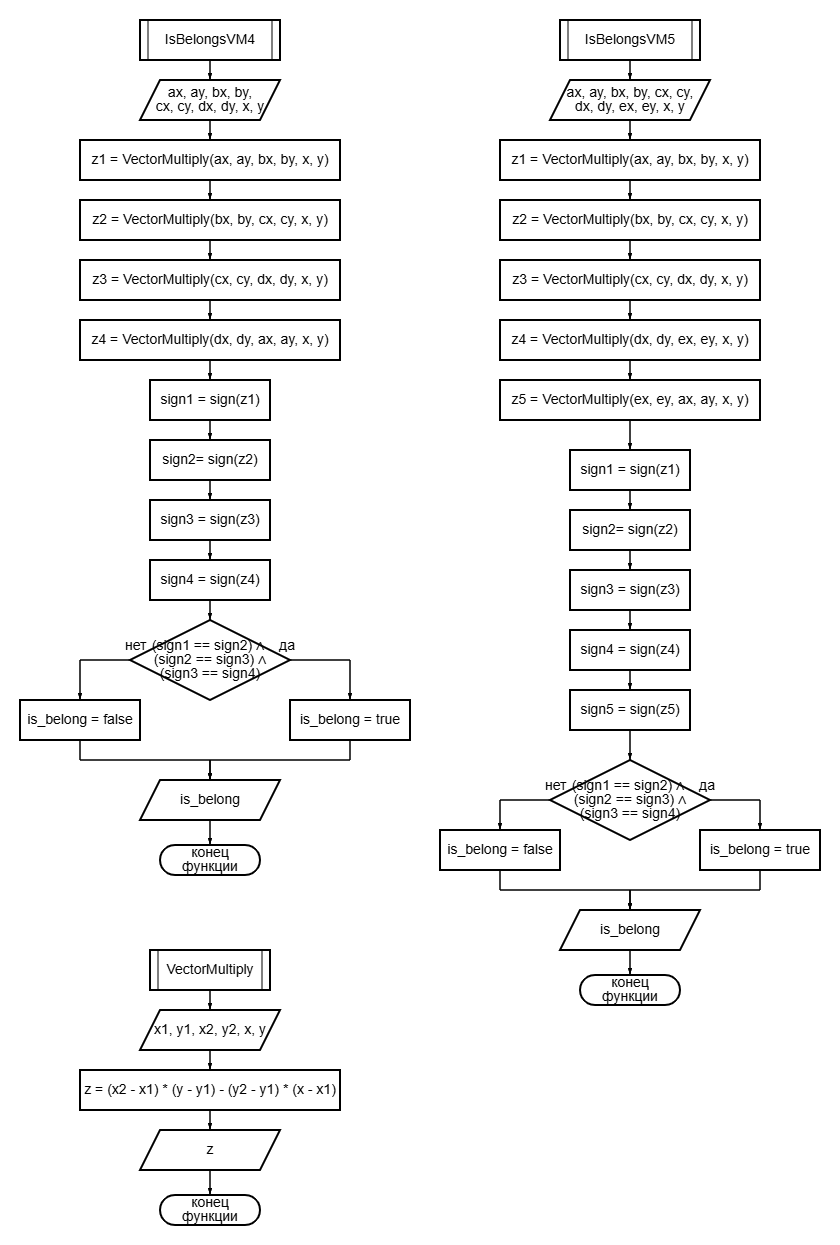
x2, y2 - координаты следующей по часовой стрелке вершины, вещественный;

x y - координаты точки, введенные пользователем вещественный;

z - результат вычисления, вещественный.

**Блок-схема:**

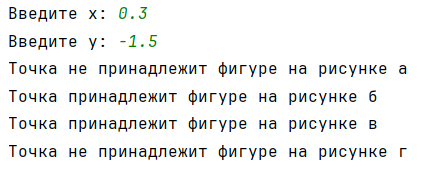
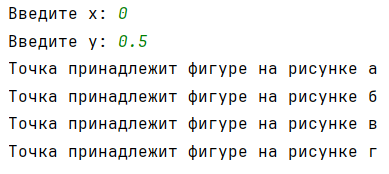
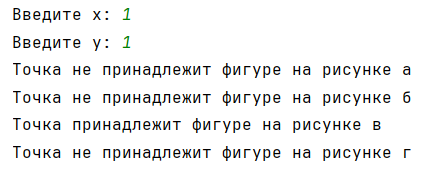




**Листинг программы:**

import kotlin.math.sign  
fun main() {  
 try {  
 *print*("Введите х: ")  
 val x = *readln*().*toDouble*()  
 *print*("Введите у: ")  
 val y = *readln*().*toDouble*()  
  
 when (*IsBelongsVM4*(-3.0, 0.0, 0.0, 1.0, 1.0, 0.0, 0.0, -1.0, x, y))  
 {  
 true -> *println*("Точка принадлежит фигуре на рисунке а")  
 else -> *println*("Точка не принадлежит фигуре на рисунке а")  
 }  
  
 when (*IsBelongsVM5*(-1.0, -2.0, -1.0, 0.0, 0.0, 2.0, 1.0, 0.0, 1.0, -2.0, x, y)){  
 true -> *println*("Точка принадлежит фигуре на рисунке б")  
 else -> *println*("Точка не принадлежит фигуре на рисунке б")  
 }  
  
 val inRect = x >= -2.0 && x <= 1.0 && y >= 0.0 && y <= 2.0  
 val distance = Math.sqrt(Math.pow(x, 2.0) + Math.pow(y + 1.0, 2.0))  
 val inCircle = distance <= 1.0 && y <= -1.0  
  
 when(inCircle || inRect){  
 true -> *println*("Точка принадлежит фигуре на рисунке в")  
 else -> *println*("Точка не принадлежит фигуре на рисунке в")  
 }  
  
 when (*IsBelongsVM4*(-2.0, 0.0, 0.0, 1.0, 1.0, 0.0, 0.0, -2.0, x, y))  
 {  
 true -> *println*("Точка принадлежит фигуре на рисунке г")  
 else -> *println*("Точка не принадлежит фигуре на рисунке г")  
 }  
  
 }  
 catch (e: Exception)  
 {  
 *println*("Неверный формат данных")  
 }  
}  
fun IsBelongsVM4(ax: Double, ay: Double, bx: Double, by: Double, cx: Double, cy: Double, dx: Double, dy: Double, x: Double, y:Double): Boolean{  
 val z1 = *VectorMultiply*(ax, ay, bx, by, x, y)  
 val z2 = *VectorMultiply*(bx, by, cx, cy, x, y)  
 val z3 = *VectorMultiply*(cx, cy, dx, dy, x, y)  
 val z4 = *VectorMultiply*(dx, dy, ax, ay, x, y)  
 val sign1 = *sign*(z1)  
 val sign2 = *sign*(z2)  
 val sign3 = *sign*(z3)  
 val sign4 = *sign*(z4)  
 val is\_belong: Boolean  
 when {  
 (sign1 == sign2) && (sign2 == sign3) && (sign3 == sign4) -> is\_belong = true  
 else -> is\_belong = false  
 }  
 return is\_belong  
}  
fun IsBelongsVM5(ax: Double, ay: Double, bx: Double, by: Double, cx: Double, cy: Double, dx: Double, dy: Double, ex: Double, ey: Double, x: Double, y:Double): Boolean{  
 val z1 = *VectorMultiply*(ax, ay, bx, by, x, y)  
 val z2 = *VectorMultiply*(bx, by, cx, cy, x, y)  
 val z3 = *VectorMultiply*(cx, cy, dx, dy, x, y)  
 val z4 = *VectorMultiply*(dx, dy, ex, ey, x, y)  
 val z5 = *VectorMultiply*(ex, ey, ax, ay, x, y)  
 val sign1 = *sign*(z1)  
 val sign2 = *sign*(z2)  
 val sign3 = *sign*(z3)  
 val sign4 = *sign*(z4)  
 val sign5 = *sign*(z5)  
 val is\_belong: Boolean  
 when {  
 (sign1 == sign2) && (sign2 == sign3) && (sign3 == sign4) && (sign4 == sign5)-> is\_belong = true  
 else -> is\_belong = false  
 }  
 return is\_belong  
}  
fun VectorMultiply(x1: Double, y1: Double, x2: Double, y2: Double, x: Double, y: Double): Double {  
 val z = (x2 - x1) \* (y - y1) - (y2 - y1) \* (x - x1)  
 return z  
}

**Тестовые ситуации:**



**Вывод:** Я научилась работать с String.format и функциями в Kotlin.