Министерство образования и молодежной политики Свердловской области



ГАПОУ СО «Екатеринбургский колледж транспортного строительства»

# Отчёт по программе «Практическое занятие 7 (задание 3)»

Выполнил: Курдин Иван

Группа: ПР-22

Преподаватель: Мирошниченко Г.В

2025

**Задание 7\_1:Вычислить площадь треугольника по формуле Герона, если заданы его стороны**

**Входные и выходные данные:**

val a = *readLine*()!!.*toDouble*()  
val b = *readLine*()!!.*toDouble*()  
val c = *readLine*()!!.*toDouble*()

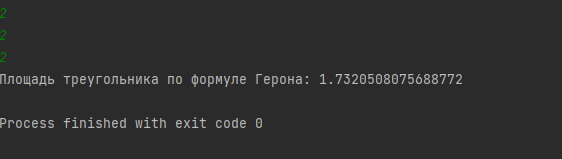
val s  
val area

**Блок-схема:**

**Листинг программы (если есть):**

import kotlin.math.sqrt  
  
fun main() {  
 try {  
 val a = *readLine*()!!.*toDouble*()  
 val b = *readLine*()!!.*toDouble*()  
 val c = *readLine*()!!.*toDouble*()  
  
 when {  
 (a + b <= c) || (a + c <= b) || (b + c <= a) -> {  
 *println*("Треугольник с такими сторонами не существует.")  
 }  
 a <= 0 || b <= 0 || c <= 0 -> {  
 *println*("Сторона треугольника не может быть меньше или равна 0.")  
 }  
 else -> {  
 val s = (a + b + c) / 2  
 val area = *sqrt*(s \* (s - a) \* (s - b) \* (s - c))  
 *println*("Площадь треугольника по формуле Герона: $area")  
 }  
 }  
 }catch (ex:Exception)  
 {  
 *println*("Ошибка")  
 }  
}

**Тестовые ситуации:**

****

----------------------------------------------------------------------------------------------------

**Задание 7\_2:Задано натуральное трехзначное число K. Выяснить, образуют ли цифры этого числа арифметическую прогрессию**

**Входные и выходные данные:**

val k

val a   
val b

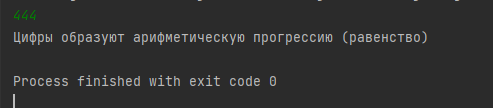
val c

**Блок-схема:**

**Листинг программы (если есть):**

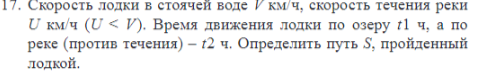
fun main() {  
 try {  
 val k = *readln*().*toInt*()  
 if (k < 100 || k > 999) {  
 *println*("Введите трехзначное число")  
 return  
 }  
 val a = k / 100  
 val b = (k % 100) / 10  
 val c = k % 10  
 when {  
 a == b && b == c -> *println*("Цифры образуют арифметическую прогрессию (равенство)")  
 (b - a) == (c - b) -> *println*("Цифры образуют арифметическую прогрессию")  
 else -> *println*("Цифры не образуют арифметическую прогрессию")  
 }  
 }catch (ex:Exception)  
 {  
 *println*("Ошибка")  
 }  
}

**Тестовые ситуации:**

****

**----------------------------------------------------------------------------------------------------**

**Задание 17\_1:**

****

**Входные и выходные данные:**

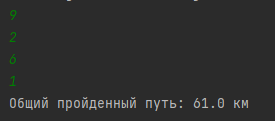
val V  
val U  
val t1  
val t2  
val Lake  
val Rive  
val Distance

**Блок-схема:**

**Листинг программы (если есть):**

fun main() {  
 try {  
 val V = *readLine*()!!.*toDouble*()  
 val U = *readLine*()!!.*toDouble*()  
 val t1 = *readLine*()!!.*toDouble*()  
 val t2 = *readLine*()!!.*toDouble*()  
 val Lake = V \* t1  
 val River = (V - U) \* t2  
 val Distance = Lake + River  
 when {  
 U >= V -> *println*("Скорость течения не может быть больше или равна скорости лодки в стоячей воде.")  
 else -> *println*("Общий пройденный путь: $Distance км")  
 }  
 }catch (ex:Exception)  
 {  
 *println*("Ошибка")  
 }  
}

**Тестовые ситуации:**

****

**----------------------------------------------------------------------------------------------------**

**Задание 17\_2:**

****

**Входные и выходные данные:**

{  
 val x1  
 val y1   
 val x2   
 val y2   
 val x3  
 val y3   
 val x4   
 val y4  
 val minX = *minOf*(x1, x2, x3, x4)  
 val maxX = *maxOf*(x1, x2, x3, x4)  
 val minY = *minOf*(y1, y2, y3, y4)  
 val maxY = *maxOf*(y1, y2, y3, y4)  
 val totalArea = (maxX - minX) \* (maxY - minY)  
 var firstQuadrantArea = 0.0

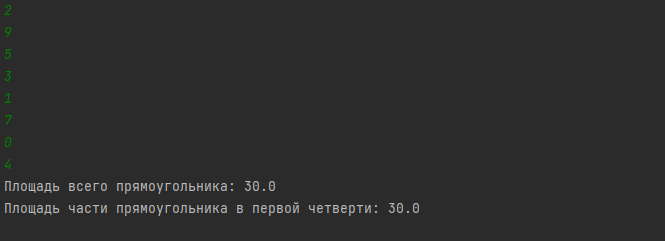
val intersectX = *maxOf*(0.0, minX)  
 val intersectY = *maxOf*(0.0, minY)

**Блок-схема:**

**Листинг программы (если есть):**

fun main() {  
 try {  
 val x1 = *readLine*()!!.*toDouble*()  
 val y1 = *readLine*()!!.*toDouble*()  
 val x2 = *readLine*()!!.*toDouble*()  
 val y2 = *readLine*()!!.*toDouble*()  
 val x3 = *readLine*()!!.*toDouble*()  
 val y3 = *readLine*()!!.*toDouble*()  
 val x4 = *readLine*()!!.*toDouble*()  
 val y4 = *readLine*()!!.*toDouble*()  
 val minX = *minOf*(x1, x2, x3, x4)  
 val maxX = *maxOf*(x1, x2, x3, x4)  
 val minY = *minOf*(y1, y2, y3, y4)  
 val maxY = *maxOf*(y1, y2, y3, y4)  
 val totalArea = (maxX - minX) \* (maxY - minY)  
 var firstQuadrantArea = 0.0  
 when {  
 maxX <= 0 || maxY <= 0 -> firstQuadrantArea = 0.0  
 minX >= 0 && minY >= 0 -> firstQuadrantArea = totalArea  
 else -> {  
 val intersectX = *maxOf*(0.0, minX)  
 val intersectY = *maxOf*(0.0, minY)  
 firstQuadrantArea = (maxX - intersectX) \* (maxY - intersectY)  
 }  
 }  
 *println*("Площадь всего прямоугольника: $totalArea")  
 *println*("Площадь части прямоугольника в первой четверти: $firstQuadrantArea")  
 }catch (ex:Exception)  
 {  
 *println*("Ошибка")  
 }  
}

**Тестовые ситуации:**

****

**Задание 20:**

****

**Входные и выходные данные:**

val r1 = *readln*()!!.*toDouble*()

val r2 = *readln*()!!.*toDouble*()

val res

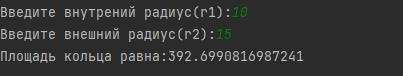
val area

**Блок-схема:**

**Листинг программы (если есть):**

fun main(){  
 try{  
 *print*("Введите внутрений радиус(r1):")  
 val r1 = *readln*()!!.*toDouble*()  
 *print*("Введите внешний радиус(r2):")  
 val r2 = *readln*()!!.*toDouble*()  
 val res = when{  
 r1 == 1.0 || r2 == 1.0 -> "Ошибка"  
 r1 < 0 || r2 < 0 -> "Ошибка"  
 r1 >= r2 -> "Ошибка"  
 else -> {  
 val area = Math.*PI* \* (r2 \* r2 - r1 \* r1)  
 "Площадь кольца равна:$area"  
 }  
 }  
 *println*(res)  
 }catch (ex:Exception)  
 {  
 *println*("Ошибка")  
 }  
}

**Тестовые ситуации:**

****

**----------------------------------------------------------------------------------------------------**