Министерство образования и молодежной политики Свердловской области



ГАПОУ СО «Екатеринбургский колледж транспортного строительства»

# Отчёт по программе «Практическое занятие 7.1.3»

Выполнил: Лежень Андрей Игоревич

Группа: ПР-23

Преподаватель: Мирошниченко Г.В

2023

**Задание:**

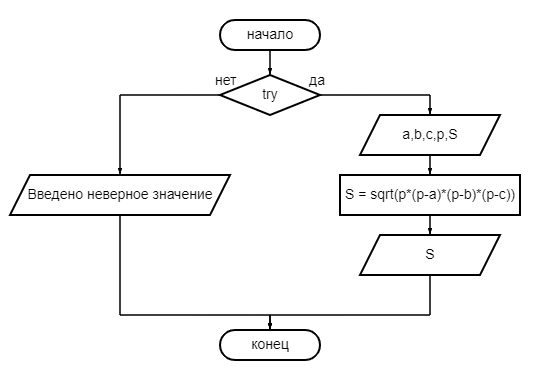
****

**Входные и выходные данные**

A,b,c – стороны треугольник, double

P – периметр треугольника, double

S – площадь треугольника

**Блок-схема**

**Тестовые ситуации**

Try{}

Catch(a: Exception){}

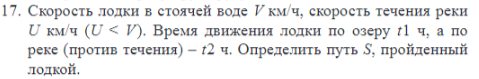
**Вывод**

Я закрепил свои знания по работе в kotlin

Листинг программы:

import kotlin.math.sqrt  
  
fun main() {  
 try  
 {  
 var a = *readln*().*toDouble*()  
 var b = *readln*().*toDouble*()  
 var c = *readln*().*toDouble*()  
 var p = (a+b+c)/2.0  
 var S = *sqrt*(p\*(p-a)\*(p-b)\*(p-c))  
 *println*("S треугольника = $S")  
 }  
 catch (a: Exception) {  
 *println*("Введено неверное значение")  
 }  
}

**Задание:**

****

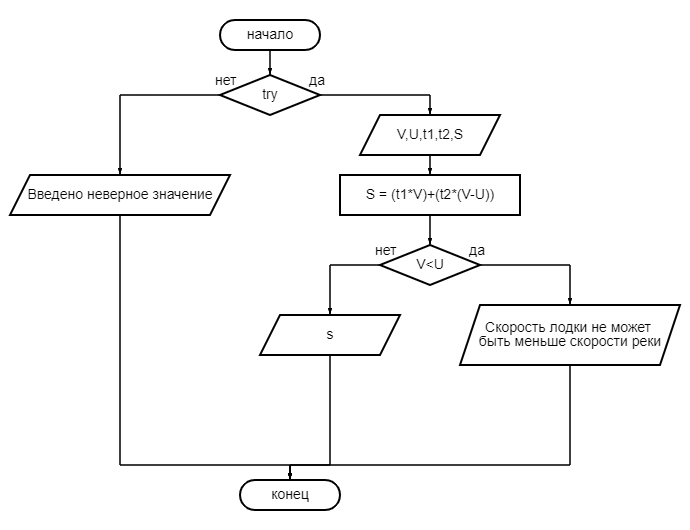
**Входные и выходные данные**

V – скорость лодки, double

U – скорость течения, double

T1, t2 – время, double

S – путь лодки, double

**Блок-схема**

**Тестовые ситуации**

Try{}

Catch(a: Exception){}

when{  
 V<U->*println*("Скорость лодки не может быть меньше скорости реки")  
 else-> *println*("Лодка прошла $S км")  
}

**Вывод**

Я закрепил свои знания по работе в kotlin

Листинг программы:

import kotlin.math.sqrt  
fun main() {  
 try  
 {  
 var V = *readln*().*toDouble*()  
 var U = *readln*().*toDouble*()  
 var t1 = *readln*().*toDouble*()  
 var t2 = *readln*().*toDouble*()  
 var S = (t1\*V)+(t2\*(V-U))  
 when{  
 V<U->*println*("Скорость лодки не может быть меньше скорости реки")  
 else-> *println*("Лодка прошла $S км")  
 }  
 }  
 catch (a: Exception) {  
 *println*("Введено неверное значение")  
 }  
}

**Задание:**

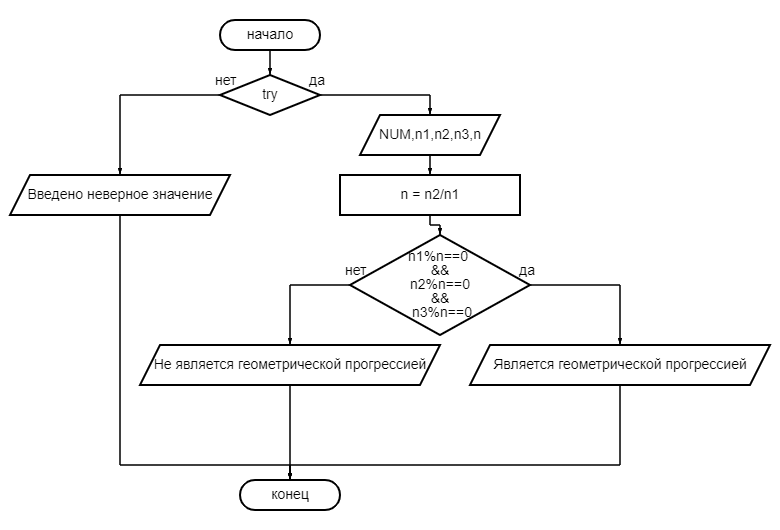
****

**Входные и выходные данные**

NUM – 3-ное число, int

N1,n2,n3 – сотри, десятки и единицы числа NUM, int

N – делитель n2 и n1, int

**Блок-схема**

**Тестовые ситуации**

Try{}

Catch(a: Exception){}

when{  
 n1%n==0 && n2%n==0 && n3%n==0-> *println*("Является геометрической прогрессией")  
 else->*println*("Не вляется геометрической прогрессией") }

**Вывод**

Я закрепил свои знания по работе в kotlin

Листинг программы:

fun main() {  
 try  
 {  
 val NUM = *readln*().*toInt*()  
 var n1 = (NUM/100)  
 var n2 = NUM%100/10  
 var n3 = NUM%10  
 var n = n2/n1  
 when{  
 n1%n==0 && n2%n==0 && n3%n==0-> *println*("Является геометрической прогрессией")  
 else->*println*("Не вляется геометрической прогрессией") }  
 }  
 catch (a: Exception) {  
 *println*("ERROR")  
 }  
}

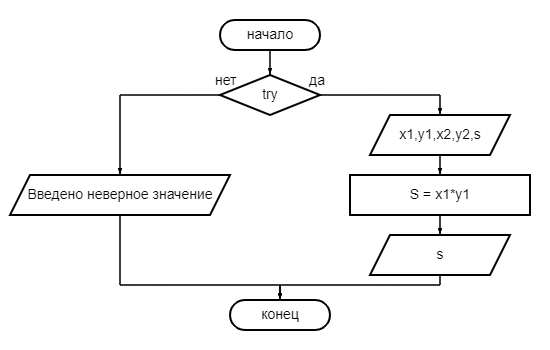
**Задание:**

****

**Входные и выходные данные**

X1,y1,x2,y2 – координаты, double

S – площадь прямоугольника в правой вершине, double

**Блок-схема**

**Тестовые ситуации**

Try{}

Catch(a: Exception){}

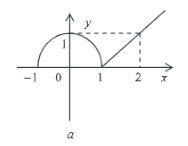
**Вывод**

Я закрепил свои знания по работе в kotlin

Листинг программы:

fun main() {  
 try  
 {  
 var x1 = *readln*().*toDouble*()  
 var y1 = *readln*().*toDouble*()  
 var x2 = *readln*().*toDouble*()  
 var y2 = *readln*().*toDouble*()  
 var S = x1\*y1  
 *println*("S = $S")  
 }  
 catch (a: Exception) {  
 *println*("ERROR")  
 }  
}

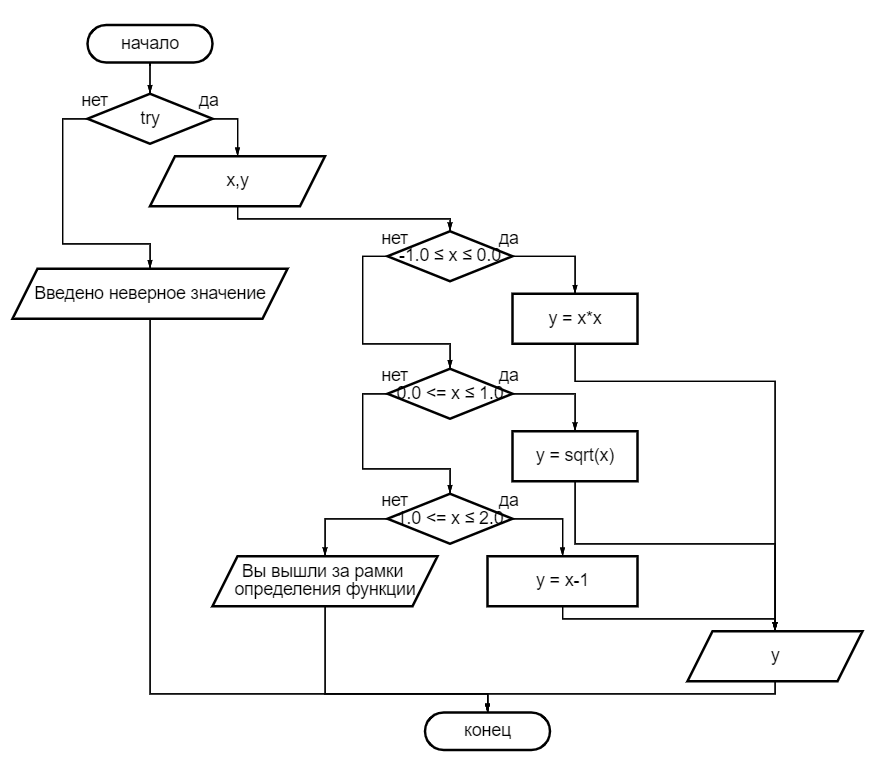
**Задание:**

**** ****

**Входные и выходные данные**

X,y – значения функции

**Блок-схема**

****

**Тестовые ситуации**

Try{}

Catch(a: Exception){}

when(x){  
 in -1.0..0.0-> y = x\*x  
 in (0.0..1.0)-> y = *sqrt*(x)  
 in (1.0..2.0)-> y = x-1  
 else-> *println*("Вы вышли за рамки определения функции")  
}

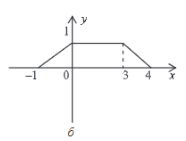
**Вывод**

Я закрепил свои знания по работе в kotlin

Листинг программы:

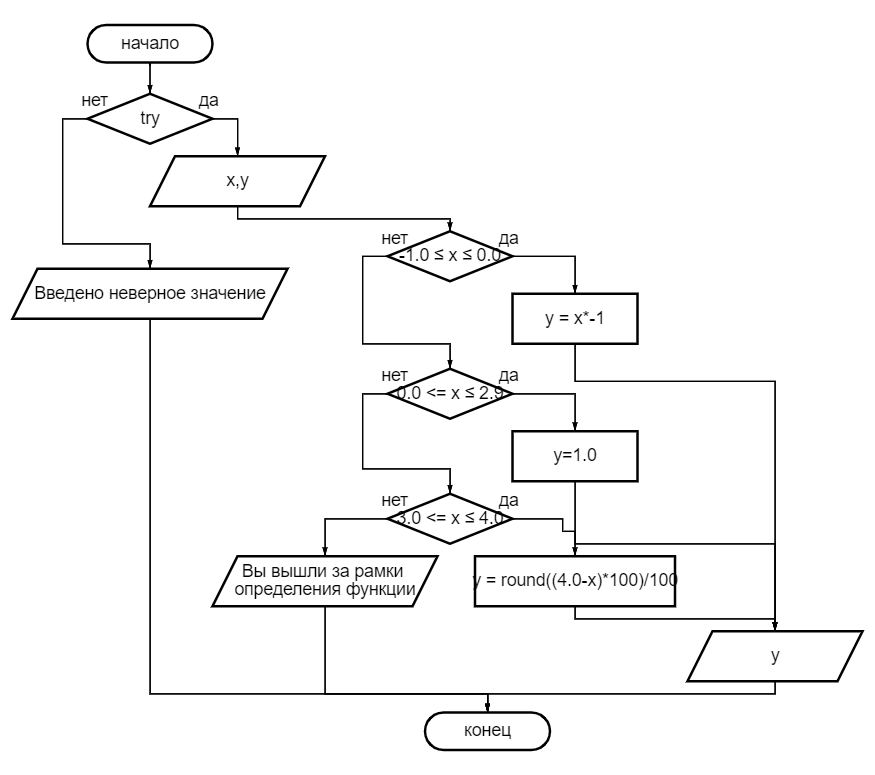
import kotlin.math.sqrt  
  
fun main() {  
 try  
 {  
 val x = *readln*().*toDouble*()  
 var y = 0.0.toDouble()  
 when(x){  
 in -1.0..0.0-> y = x\*x  
 in (0.0..1.0)-> y = *sqrt*(x)  
 in (1.0..2.0)-> y = x-1  
 else-> *println*("Вы вышли за рамки определения функции")  
 }  
 *println*("y = $y")  
 }  
 catch (a: Exception) {  
 *println*("ERROR")  
 }  
}

**Задание:**

**** 

**Входные и выходные данные**

X,y – значения функции

**Блок-схем**

**Тестовые ситуации**

Try{}

Catch(a: Exception){}

when(x){  
 in -1.0..0.0-> y = x\*-1  
 in (0.0..2.9)-> y = 1.0  
 in (3.0..4.0)-> y = *round*((4.0-x)\*100)/100  
 else-> *println*("Вы вышли за рамки определения функции")  
}

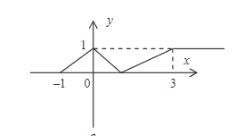
**Вывод**

Я закрепил свои знания по работе в kotlin

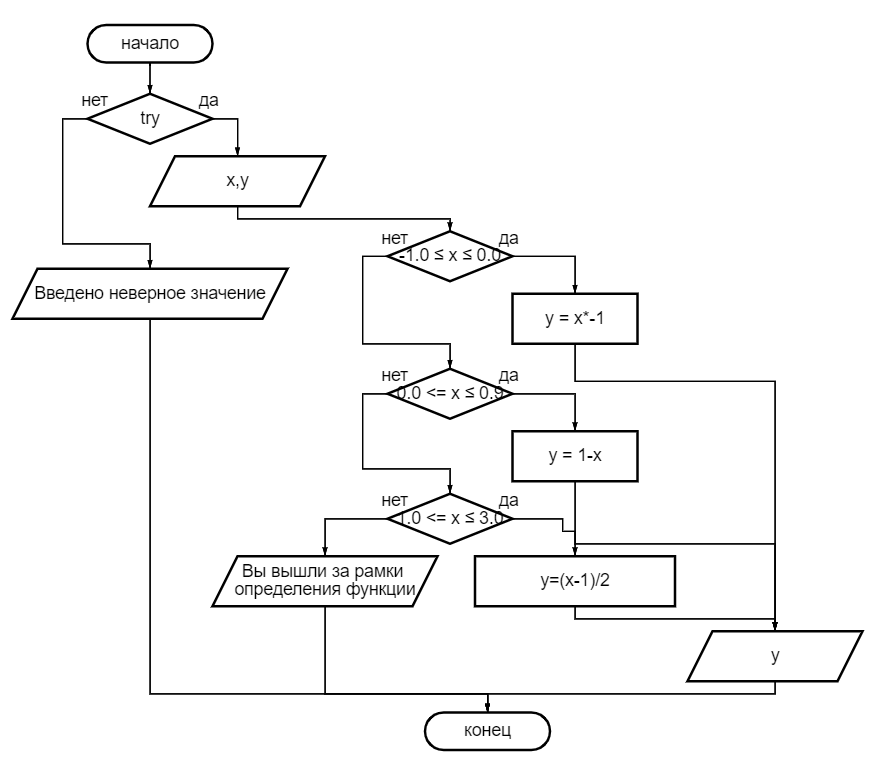
Листинг программы:

import kotlin.math.round  
import kotlin.math.sqrt  
  
fun main() {  
 try  
 {  
 val x = *readln*().*toDouble*()  
 var y = 0.0.toDouble()  
 when(x){  
 in -1.0..0.0-> y = x\*-1  
 in (0.0..2.9)-> y = 1.0  
 in (3.0..4.0)-> y = *round*((4.0-x)\*100)/100  
 else-> *println*("Вы вышли за рамки определения функции")  
 }  
 *println*("y = $y")  
 }  
 catch (a: Exception) {  
 *println*("ERROR")  
 }  
}

**Задание:**

**** **Входные и выходные данные**

X,y – значения функции

**Блок-схем**

**Тестовые ситуации**

Try{}

Catch(a: Exception){}

when(x){  
 in -1.0..0.0-> y = x\*-1  
 in (0.0..0.9)-> y = 1-x  
 in (1.0..3.0)-> y = (x-1)/2  
 else-> *println*("Вы вышли за рамки определения функции")  
}

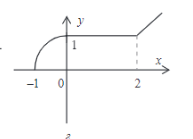
**Вывод**

Я закрепил свои знания по работе в kotlin

Листинг программы:

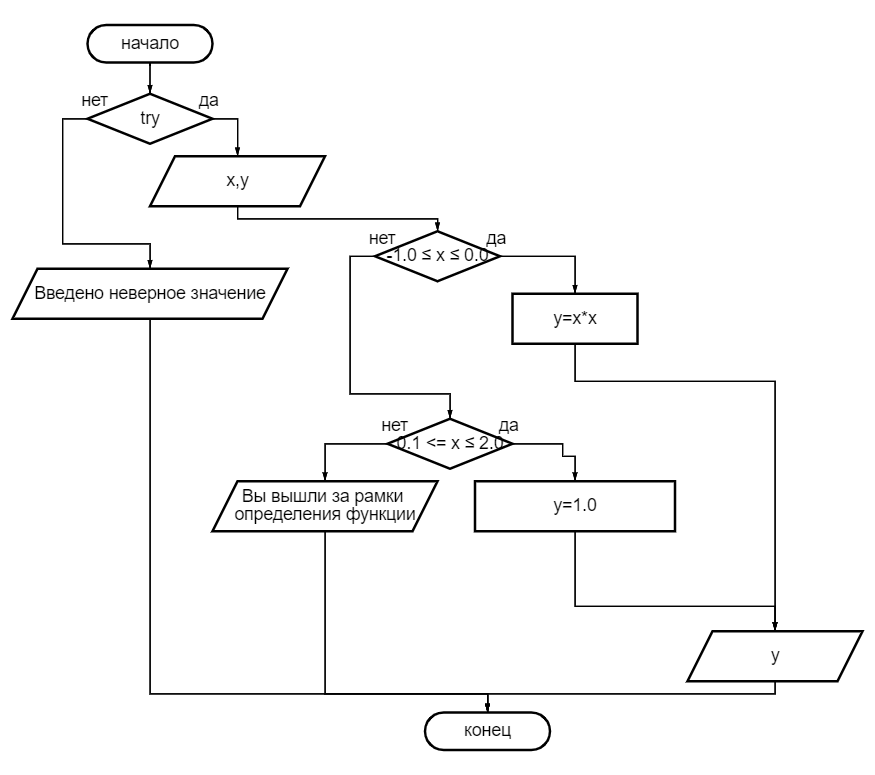
fun main() {  
 try  
 {  
 val x = *readln*().*toDouble*()  
 var y = 0.0.toDouble()  
 when(x){  
 in -1.0..0.0-> y = x\*-1  
 in (0.0..0.9)-> y = 1-x  
 in (1.0..3.0)-> y = (x-1)/2  
 else-> *println*("Вы вышли за рамки определения функции")  
 }  
 *println*("y = $y")  
 }  
 catch (a: Exception) {  
 *println*("ERROR")  
 }  
}

**Задание:**

**** 

**Входные и выходные данные**

X,y – значения функции

**Блок-схем**

**Тестовые ситуации**

Try{}

Catch(a: Exception){}

when(x){  
 in -1.0..0.0-> y = x\*x  
 in (0.0..2.0)-> y = 1.0  
 else-> *println*("Вы вышли за рамки определения функции")  
}

**Вывод**

Я закрепил свои знания по работе в kotlin

Листинг программы:

fun main() {  
 try  
 {  
 val x = *readln*().*toDouble*()  
 var y = 0.0.toDouble()  
 when(x){  
 in -1.0..0.0-> y = x\*x  
 in (0.0..2.0)-> y = 1.0  
 else-> *println*("Вы вышли за рамки определения функции")  
 }  
 *println*("y = $y")  
 }  
 catch (a: Exception) {  
 *println*("ERROR")  
 }  
}

**ч**