Министерство образования и молодежной политики Свердловской области



ГАПОУ СО «Екатеринбургский колледж транспортного строительства»

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

Выполнил: Ялымов Пётр Васильевич

Группа: ПР-22

Преподаватель: Мирошниченко Г.В

2024

**Задание: 8.**

Определить вершины параболы y=ax^2+bx+c (a!=0). Коэффициенты a,b,c ввести.

**Входные и выходные данные**

**a**-входной, double

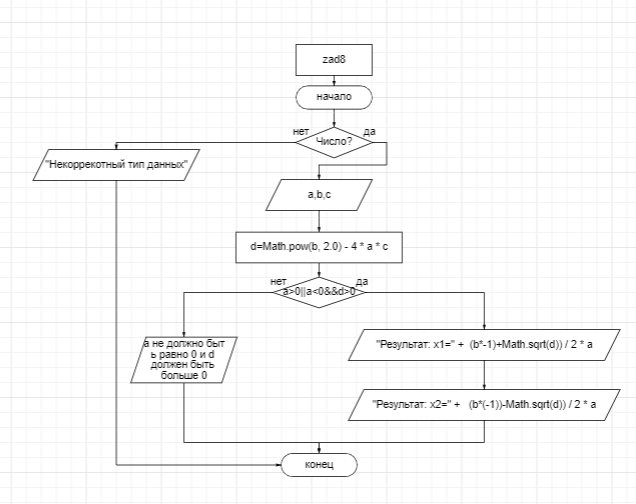
**b**-входной, double

**c**-входной, double

**x1**-выходной, double

**x2**-выходной, double

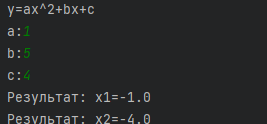
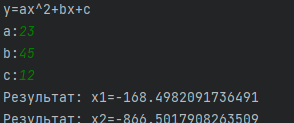
**Блок-схема**

****

**Листинг программы**

fun main()  
{  
 try  
 {  
 *println*("y=ax^2+bx+c")  
 *print*("a:")  
 var a= *readLine*()!!.*toDouble*()  
 *print*("b:")  
 var b= *readLine*()!!.*toDouble*()  
 *print*("c:")  
 var c= *readLine*()!!.*toDouble*()  
 var d=Math.pow(b, 2.0) - 4 \* a \* c  
 when(a>0||a<0&&d>0)  
 {  
 true->  
 {  
 *println*("Результат: x1=" + ((b\*-1)+Math.sqrt(d)) / 2 \* a)  
 *print*("Результат: x2=" + ((b\*(-1))-Math.sqrt(d)) / 2 \* a)  
 }  
 false-> *println*("a не должно быть равно 0 и d должен быть больше 0")  
 }  
 }catch (e:Exception)  
 {  
 *print*("Некоррекотный тип данных")  
 }  
}

**Тестовые ситуации**

**** ****

**Задание: 18.**

****

**Входные и выходные данные**

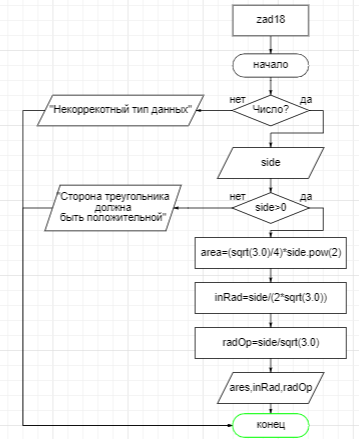
**side**-входной, double

**area**-выходной, double, площадь треугольника

**inRad**-выходной, double, радиус вписанной окружности

**radOp**-выходной, double, радиус описанной окружности

**Блок-схема**

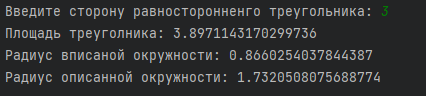
****

**Листинг программы**

import kotlin.math.pow  
import kotlin.math.sqrt  
  
fun main()  
{  
 try  
 {  
 *print*("Введите сторону равносторонненго треугольника: ")  
 var side= *readLine*()!!.*toDouble*()  
 when(side>0)  
 {  
 true->  
 {  
 var area=(*sqrt*(3.0)/4)\*side.*pow*(2)  
 var inRad=side/(2\**sqrt*(3.0))  
 var radOp=side/*sqrt*(3.0)  
 *println*("Площадь треуголника: $area")  
 *println*("Радиус вписаной окружности: $inRad")  
 *println*("Радиус описанной окружности: $radOp")  
 }  
 false->*println*("Сторона треугольника должна быть положительной")  
 }  
 }catch (e:Exception)  
 {  
 *print*("Некоррекотный тип данных")  
 }  
}

}

**Тестовые ситуации**



**Задание: 8.2**

****

**Входные и выходные данные**

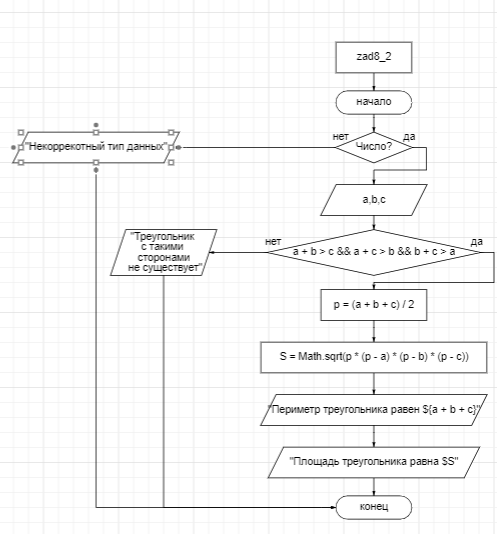
**a**-входной, double, сторона треугольника

**b**-входной, double, сторона треугольника

**с**-входной, double, сторона треугольника

**S**-выходной, double, площадь треугольника

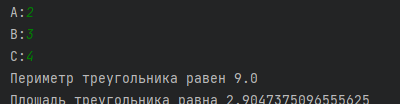
**Блок-схема**

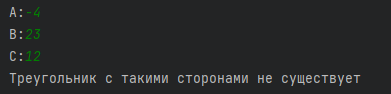
****

**Листинг программы**

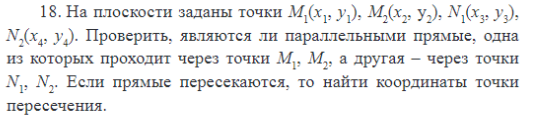
fun main()  
{  
 try  
 {  
 *print*("A:")  
 val a = *readLine*()!!.*toDouble*()  
 *print*("B:")  
 val b = *readLine*()!!.*toDouble*()  
 *print*("C:")  
 val c = *readLine*()!!.*toDouble*()  
 when (a + b > c && a + c > b && b + c > a)  
 {  
 true->  
 {  
 val p = (a + b + c) / 2  
 val S = Math.sqrt(p \* (p - a) \* (p - b) \* (p - c))  
 *println*("Периметр треугольника равен ${a + b + c}")  
 *println*("Площадь треугольника равна $S")  
 }  
 false->*println*("Треугольник с такими сторонами не существует")  
 }  
 }catch (e:Exception)  
 {  
 *print*("Некоррекотный тип данных")  
 }  
}

**Тестовые ситуации**





**Задание: 18.2**

****

**Входные и выходные данные**

**X1**-входной, double

**X2**-входной, double

**X3**-входной, double

**X4**-входной, double

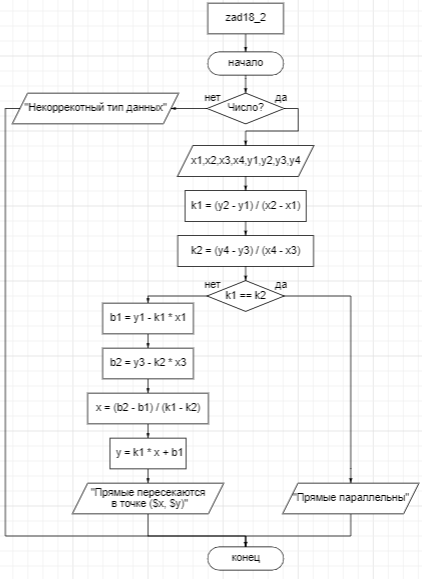
**Y1**-входной, double

**Y2**-входной, double

**Y3**-входной, double

**Y4**-входной, double

**Блок-схема**

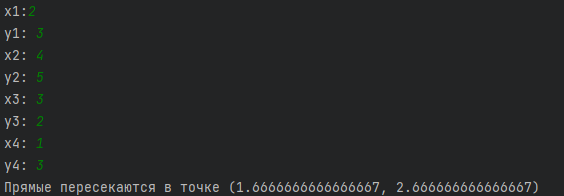
****

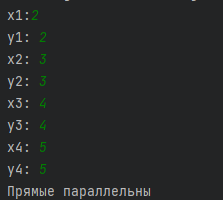
**Листинг программы**

fun main()  
{  
 try  
 {  
 *print*("x1:")  
 val x1 = *readLine*()!!.*toDouble*()  
 *print*("y1: ")  
 val y1 = *readLine*()!!.*toDouble*()  
 *print*("x2: ")  
 val x2 = *readLine*()!!.*toDouble*()  
 *print*("y2: ")  
 val y2 = *readLine*()!!.*toDouble*()  
 *print*("x3: ")  
 val x3 = *readLine*()!!.*toDouble*()  
 *print*("y3: ")  
 val y3 = *readLine*()!!.*toDouble*()  
 *print*("x4: ")  
 val x4 = *readLine*()!!.*toDouble*()  
 *print*("y4: ")  
 val y4 = *readLine*()!!.*toDouble*()  
 val k1 = (y2 - y1) / (x2 - x1)  
 val k2 = (y4 - y3) / (x4 - x3)  
  
 when (k1 == k2) {  
 true -> *println*("Прямые параллельны")  
 false -> {  
 val b1 = y1 - k1 \* x1  
 val b2 = y3 - k2 \* x3  
 val x = (b2 - b1) / (k1 - k2)  
 val y = k1 \* x + b1  
 *println*("Прямые пересекаются в точке ($x, $y)")  
 }  
 }  
 }catch (e:Exception)  
 {  
 *print*("Некоррекотный тип данных")  
 }  
}

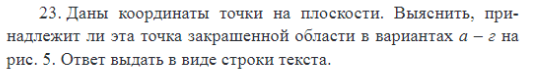
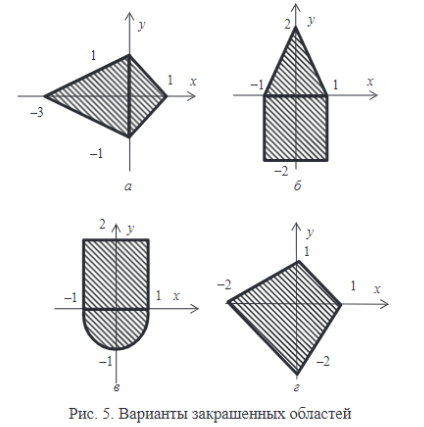
**Тестовые ситуации**







**Задание: 23**

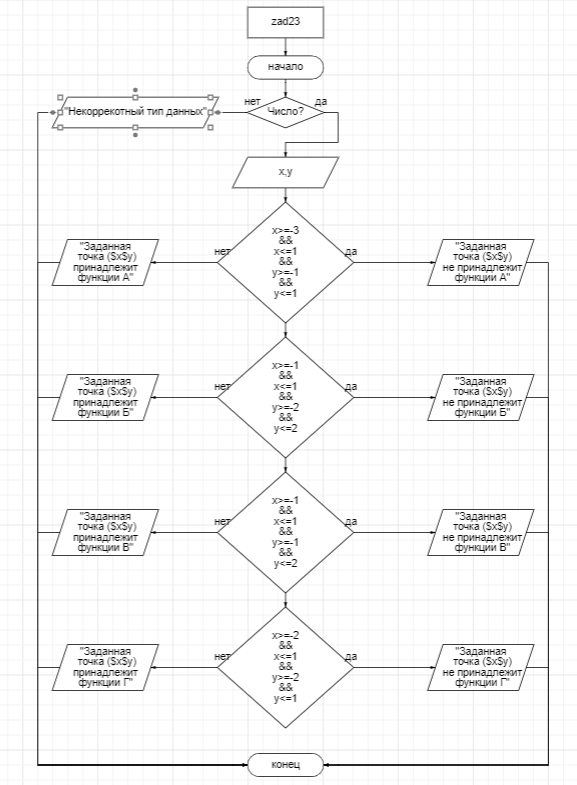
**** ****

**Входные и выходные данные**

**X1**-входной, double

**Y1**-входной, double

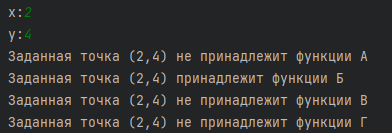
**Блок-схема**

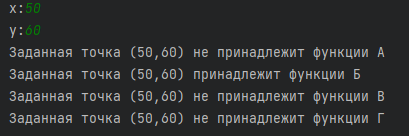
****

**Листинг программы**

fun main()  
{  
 try  
 {  
 *print*("x:")  
 var x=*readLine*()!!.*toInt*()  
 *print*("y:")  
 var y=*readLine*()!!.*toInt*()  
 when(x>=-3&&x<=1&&y>=-1&&y<=1)  
 {  
 true-> *println*("Заданная точка ($x$y) принадлежит функции А")  
 false->*println*("Заданная точка ($x$y) не принадлежит функции А")  
 }  
 when(x>=-1&&x<=1&&y>=-2&&y<=2)  
 {  
 true-> *println*("Заданная точка ($x$y) принадлежит функции Б")  
 false->*println*("Заданная точка ($x$y) принадлежит функции Б")  
 }  
 when(x>=-1&&x<=1&&y>=-1&&y<=2)  
 {  
 true->*println*("Заданная точка ($x$y) принадлежит функции В")  
 false->*println*("Заданная точка ($x$y) не принадлежит функции В")  
 }  
 when(x>=-2&&x<=1&&y>=-2&&y<=1)  
 {  
 true-> *println*("Заданная точка ($x$y) принадлежит функции Г")  
 false->*println*("Заданная точка ($x$y) не принадлежит функции Г")  
 }  
 }catch (e:Exception)  
 {  
 *print*("Некоррекотный тип данных")  
 }  
}

**Тестовые ситуации**

****

****

