Министерство образования и молодежной политики Свердловской области



ГАПОУ СО «Екатеринбургский колледж транспортного строительства»

Отчёт по программе «**Практическое занятие 7.2**»

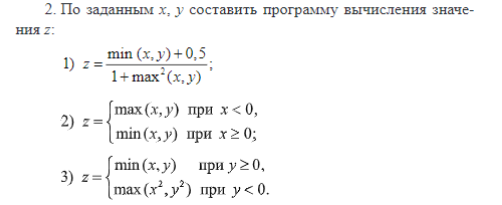
Выполнил: Горовой В.С

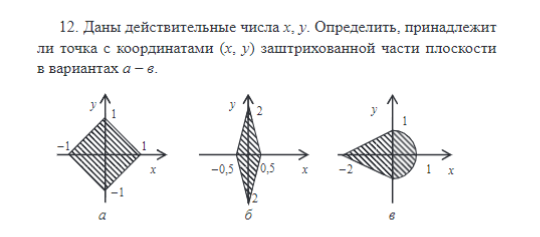
Группа: ПР-22

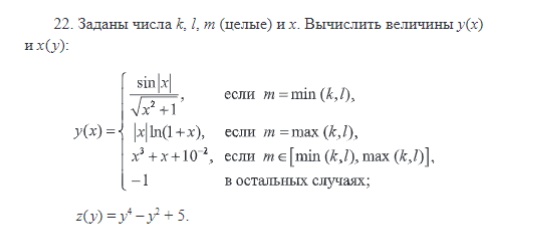
Преподаватель: Мирошниченко Г.В

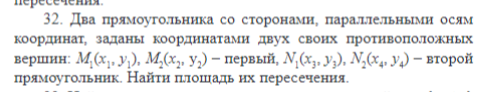
2023

**Задание:**

****

****

****

****

****

**Входные и выходные данные**

**2. Входные данные:**

**X,Y – точки-координаты, double, число с плавающей запятой**

**Выходные данные:**

**Z – функция, double, число с плавающей запятой.**

**12. Входные данные:**

**X,Y – точки-координаты, double, число с плавающей запятой**

**Выходные данные:**

**Текст: “точка принадлежит/не принадлежит окружности N”**

**22. Входные данные:**

**K,L,M – переменные, int, целочисленный тип**

**X – значение в формуле, double, целочисленный тип**

**Выходные данные:**

**Y – формула y(x), double, число с плавающей запятой.**

**xY – формула x(y), double, число с плавающей запятой.**

**32. Входные данные:**

**mX1 mY1 mX2 mY2 – точки-координаты треугольника M, double, число с плавающей запятой  
nX1 nY1 nX2 nY2 – точки-координаты треугольника N, double, число с плавающей запятой**

**Выходные данные:**

**Peresechenie – площадь пересечения двух прямоугольников, double, число с плавающей запятой**

**42. Входные данные:**

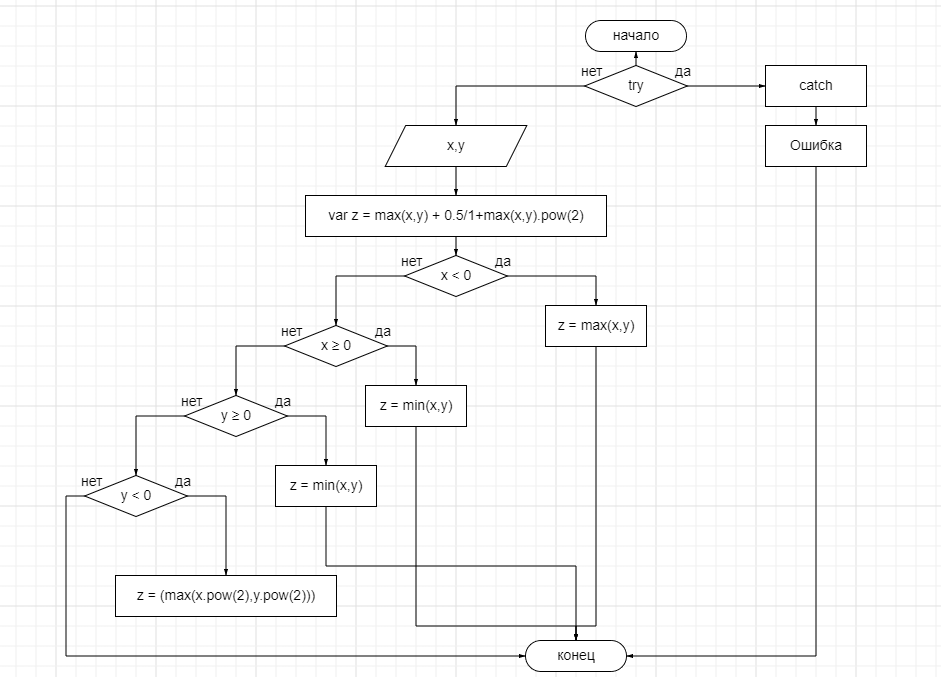
**A,B,C – переменные формулы ax2+bx2+c, double, число с плавающей запятой.**

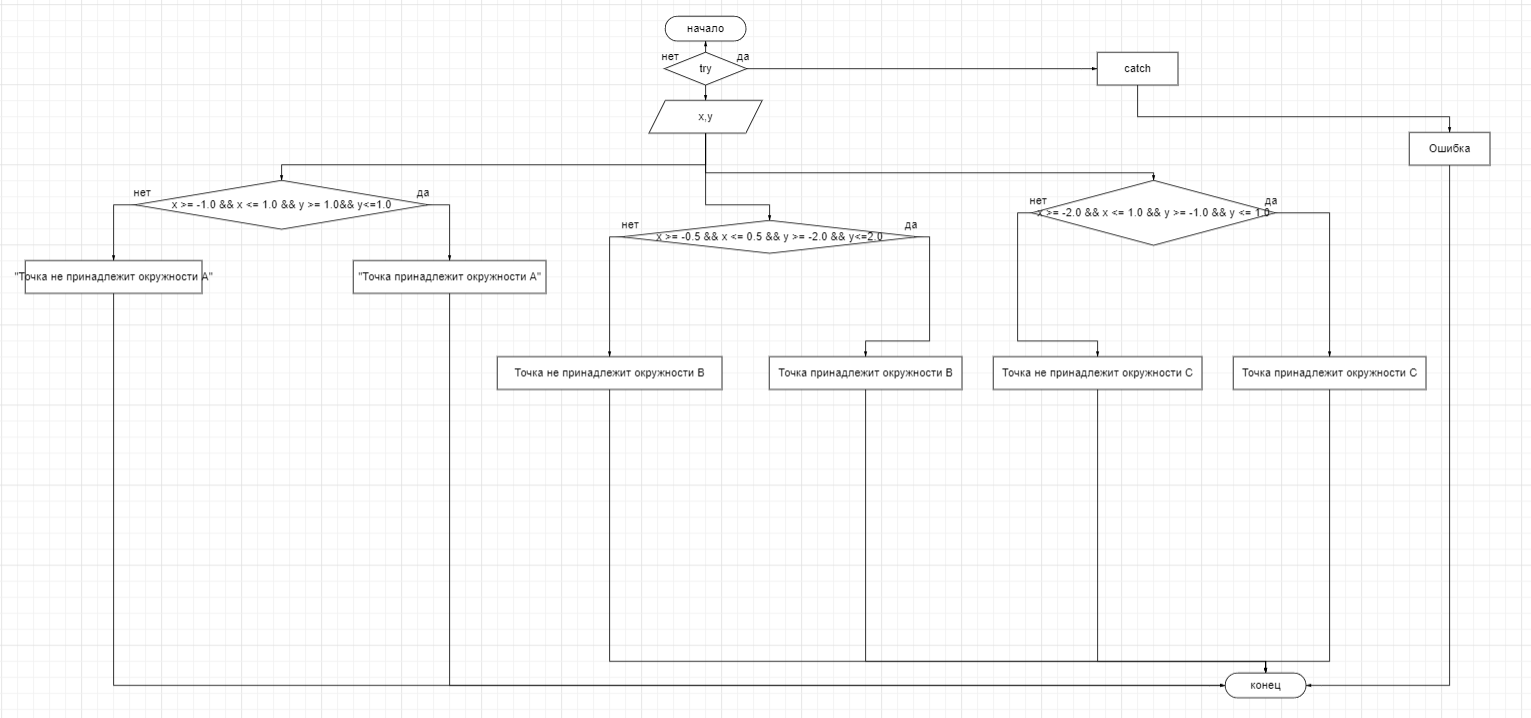
**Выходные данные:**

**X1 – корень уравнения, double, число с плавающей запятой.  
X2 - корень уравнения, double, число с плавающей запятой.**

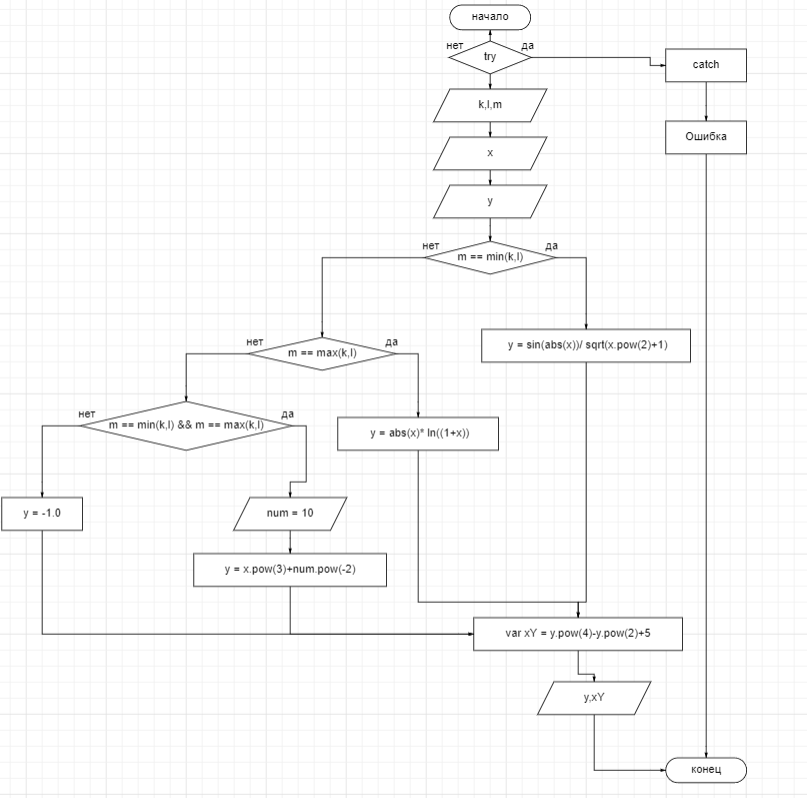
**X - корень уравнения (если уравнение имеет один корень), double, число с плавающей запятой.**

**Блок-схема**

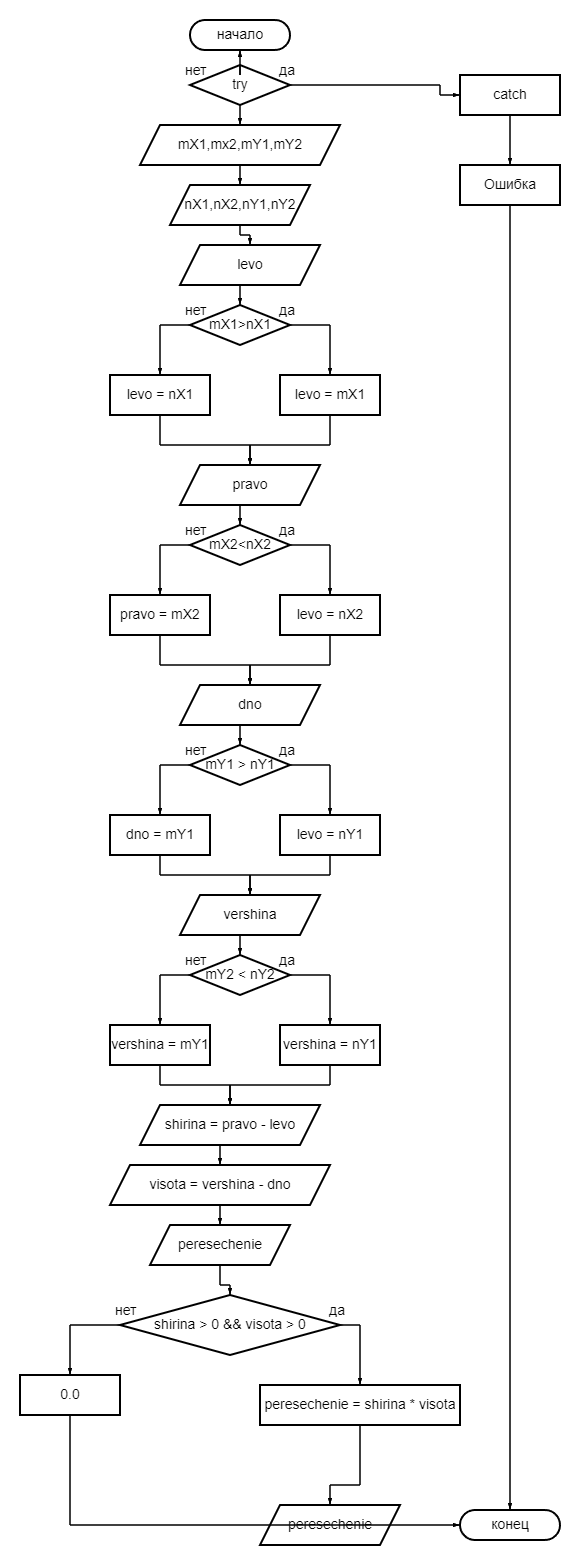
**2. **

**12.  
**

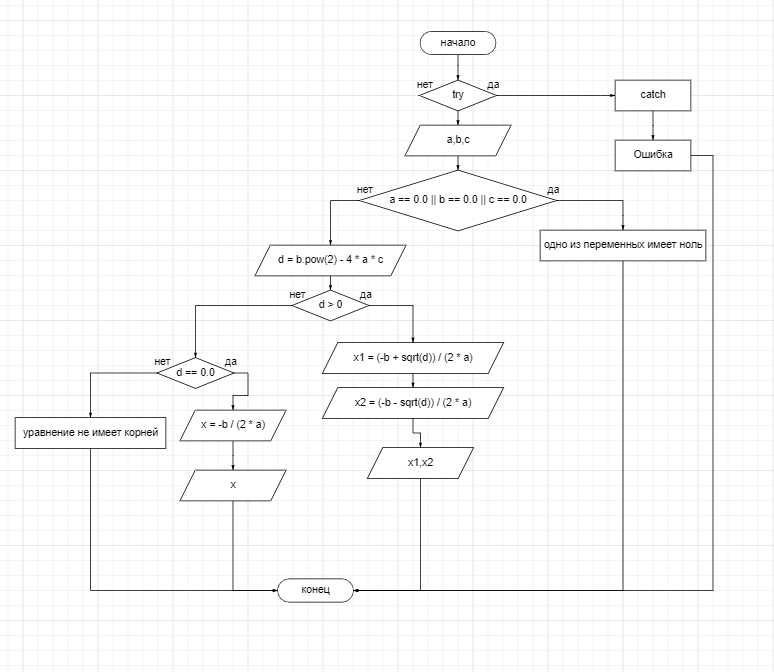
**22.**

****

**32.**

****

**42.**

****

**Листинг программы (если есть)**

**2.**

import kotlin.math.min  
import kotlin.math.max  
import kotlin.math.pow  
fun main(){  
 try  
 {  
 var x = *readLine*()!!.*toDouble*()  
 var y = *readLine*()!!.*toDouble*()  
  
 var z = *max*(x,y) + 0.5/1+*max*(x,y).*pow*(2)  
  
 when  
 {  
 (x<0) -> z = *max*(x,y)  
 (x>=0) -> z = *min*(x,y)  
 (y >= 0) -> z = *min*(x,y)  
 (y < 0) -> z = (*max*(x.*pow*(2),y.*pow*(2)))  
 }  
 *println*(z);  
 }  
 catch (e:Exception)  
 {  
 *println*("Ошибка.")  
 }  
}

**12.**

fun main(){  
 try  
 {  
 *println*("Введите x и y: ")  
 val x = *readLine*()!!.*toDouble*()  
 val y = *readLine*()!!.*toDouble*()  
 if (x >= -1.0 && x <= 1.0 && y >= 1.0&& y<=1.0)  
 {  
 *println*("Точка принадлежит окружности А")  
 }  
 else  
 {  
 *println*("Точка не принадлежит окружности А")  
 }  
 if (x >= -0.5 && x <= 0.5 && y >= -2.0 && y<=2.0)  
 {  
 *println*("Точка принадлежит окружности B")  
 }  
 else  
 {  
 *println*("Точка не принадлежит окружности B")  
 }  
 if (x >= -2.0 && x <= 1.0 && y >= -1.0 && y <= 1.0)  
 {  
 *println*("Точка принадлежит окружности C")  
 }  
 else  
 {  
 *println*("Точка не принадлежит окружности C")  
 }  
  
  
 }  
 catch (e:Exception)  
 {  
 *println*("Ошибка")  
 }  
}

**22.**

import kotlin.math.\*  
  
fun main(){  
try {  
 *println*("Введите k,l,m,x")  
 var k = *readLine*()!!.*toInt*()  
 var l = *readLine*()!!.*toInt*()  
 var m = *readLine*()!!.*toInt*()  
 var x = *readLine*()!!.*toDouble*()  
  
 var y = 0.0  
 if (m == *min*(k,l))  
 {  
 y = *sin*(*abs*(x))/ *sqrt*(x.*pow*(2)+1)  
 }  
 else if (m == *max*(k,l))  
 {  
 y = *abs*(x)\* *ln*((1+x))  
 }  
 else if (m == *min*(k,l) && m == *max*(k,l))  
 {  
 var num = 10.toDouble()  
 y = x.*pow*(3)+num.*pow*(-2)  
 }  
 else  
 {  
 y = -1.0  
 }  
  
 var xY = y.*pow*(4)-y.*pow*(2)+5  
 *println*("y(x): $y")  
 *println*("x(y): $xY")  
  
}  
catch (e:Exception)  
{  
 *println*("Ошибка")  
}  
}

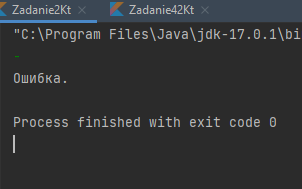
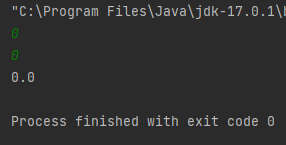
**32.**

fun main(){  
 try {  
 *println*("Введите mX1,mX2,mY1,mY2:")  
 var mX1 = *readLine*()!!.*toDouble*()  
 var mY1 = *readLine*()!!.*toDouble*()  
 var mX2 = *readLine*()!!.*toDouble*()  
 var mY2 = *readLine*()!!.*toDouble*()  
 *println*("Введите nX1,nX2,nY1,nY2:")  
 var nX1 = *readLine*()!!.*toDouble*()  
 var nY1 = *readLine*()!!.*toDouble*()  
 var nX2 = *readLine*()!!.*toDouble*()  
 var nY2 = *readLine*()!!.*toDouble*()  
  
 val levo = if (mX1 > nX1) mX1 else nX1  
 val pravo = if (mX2 < nX2) mX2 else nX2  
 val dno = if (mY1 > nY1) mY1 else nY1  
 val vershina = if (mY2 < nY2) mY2 else nY2  
  
 val shirina = pravo - levo  
 val visota = vershina - dno  
  
 val peresechenie = if (shirina > 0 && visota > 0) shirina \* visota else 0.0  
  
 *println*("Площадь пересечения двух прямоугольников: $peresechenie")  
 }  
 catch (e:Exception)  
 {  
 *println*("Ошибка")  
 }  
}

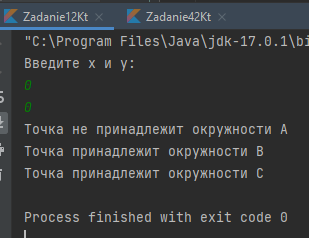
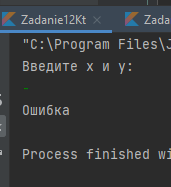
**42.**

import kotlin.math.sqrt  
import kotlin.math.pow  
  
fun main() {  
 try {  
 *println*("Введите A,B,C: ")  
 var a = *readLine*()!!.*toDouble*()  
 var b = *readLine*()!!.*toDouble*()  
 var c = *readLine*()!!.*toDouble*()  
  
 if (a == 0.0 || b == 0.0 || c == 0.0)  
 {  
 *println*("одно из переменных имеет ноль")  
 }  
 else  
 {  
 val d = b.*pow*(2) - 4 \* a \* c  
  
 if (d > 0) {  
 val x1 = (-b + *sqrt*(d)) / (2 \* a)  
 val x2 = (-b - *sqrt*(d)) / (2 \* a)  
 *println*("уравнение имеет два корня: $x1, $x2")  
 }  
 else if (d == 0.0)  
 {  
 val x = -b / (2 \* a)  
 *println*("уравнение имеет один корень: $x")  
 }  
 else  
 {  
 *println*("уравнение не имеет корней")  
 }  
 }  
 }  
 catch (e:Exception)  
 {  
 *println*("Ошибка")  
 }  
  
}

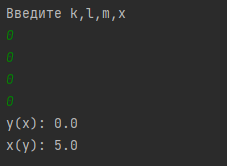
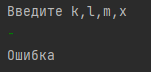
**Тестовые ситуации**

**2.** ****

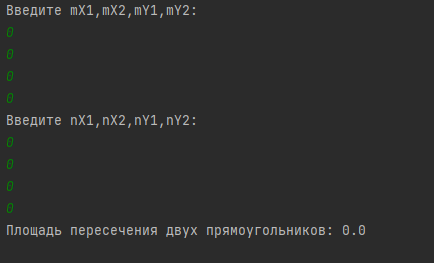
**12.**



22.



32.



42.

