ММинистерство образования и молодежной политики Свердловской области



ГАПОУ СО «Екатеринбургский колледж транспортного строительства»

# Отчёт по программе «Практическое занятие 12 (корутины и потоки)»

Выполнил: Мартынов Арсений

Группа: ПР-21

Преподаватель: Мирошниченко Г.В

2024

**Задание:**

1.Реализовать через корутины вызов функций нажатий и

параллельного получения данных из n функций (n вводит пользователь).

Затем, нужно их объединять и отображать.

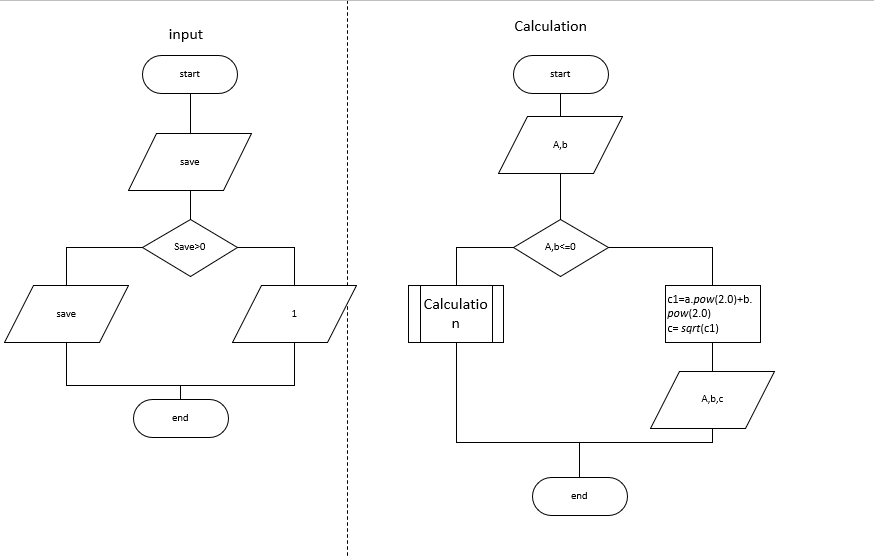
**Входные и выходные данные**

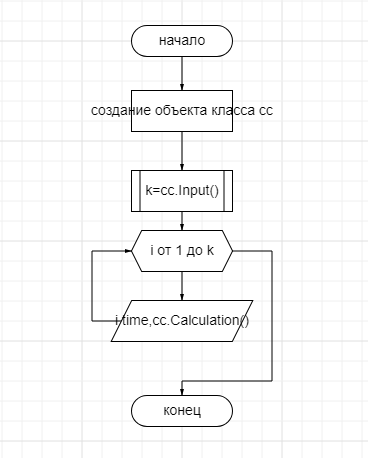
Класс class

**Выходные**

Строка и переменные c,a,b(хранят в себе значение катетов и гипотенузы)

**Блок-схема**

****

****

**Листинг программы**

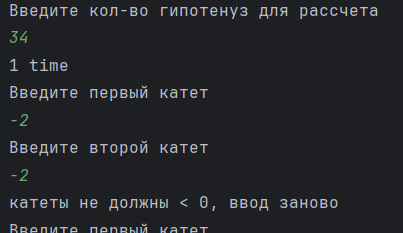
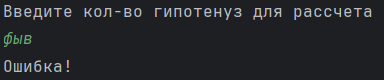
MAIN

suspend fun main(){  
 try {  
  
 var cc:`class`=`class`()  
 var k=cc.Input()  
 GlobalScope.*launch* **{** for(i in 1..k){  
 *println*("$i time")  
 *println*(cc.Calculation())  
 }  
 **}** *runBlocking* **{** delay(300000) **}** }catch (e:Exception){  
 *println*("Ошибка!")  
 }  
   
}

CLASS

import kotlin.math.\*  
open class `class`(){  
 fun Input():Int {  
 *println*("Введите кол-во гипотенуз для рассчета")  
 var Save=*readln*()!!.*toInt*()  
 if(Save>0)  
 return Save  
 else {  
 *println*("число должно быть > 0, теперь оно 1")  
 return 1  
 }  
 }  
  
 fun Calculation():String  
 {  
 *println*("Введите первый катет"); var a=*readln*()!!.*toDouble*()  
 *println*("Введите второй катет"); var b=*readln*()!!.*toDouble*()  
 if(a<=0||b<=0){*println*("катеты не должны < 0, ввод заново");Calculation()}  
 else{  
 var c:Double  
 var c1=a.*pow*(2.0)+b.*pow*(2.0)  
 c= *sqrt*(c1)  
 return "катет 1:$a катет 2:$b гипотенуза:$c"  
 }  
 return ""  
 }  
}

**Тестовые ситуации**

****

**Задача**

2.Создайте программу для загрузки участников для всех репозиториев

данной организации, с сортировкой пользователей по количеству их

репрозиториев.

Вводится имя пользователя и токен (или пароль) GitHub в соответствующие

поля. Если в раскрывающемся меню вариантов выбран вариант

«БЛОКИРОВКА», нажмите «Загрузить участников». Пользовательский

интерфейс должен зависнуть на некоторое время, а затем отобразить

отсортированный список участников с указанием количества репрозиториев .

**Входные**

Log,pas,number,open

**Выходные данные**

элементы листа

**Блок-схема**

**Листинг программы**

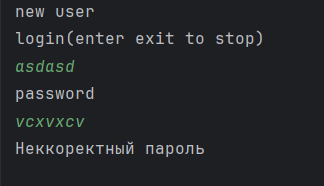
CLASS

import kotlinx.coroutines.delay  
open class zad2(var log:String, var pass:String, var number:Int, var open:Boolean) {  
  
 suspend fun SortedRep(reposit: List<zad2>): List<zad2> {  
 delay(2000L)  
 return reposit.*filter* **{ it**.open **}**.*sortedWith*(*compareBy***{ it**.number**}**)  
 }  
  
 suspend fun Get\_info():String{  
 delay(1000)  
 return "LOGIN:$log,Password:$pass,Number:$number,"  
 }  
}

MAIN

suspend fun main()= *runBlocking***{** var reposit=*mutableListOf*<zad2>()  
 var st=true  
 while (st==true) {  
 try {  
 *println*("new user")  
 *println*("login(enter exit to stop)")  
 var login = *readln*()  
 if (login == "exit") break  
 else{  
 *println*("password")  
 var pas = *readln*()  
 if(pas.length<8) {*println*("Неккоректный пароль");return@runBlocking}  
 *println*("number")  
 var number = *readln*()!!.*toInt*()  
 if(number<0) {*println*("Неккоректный пароль");return@runBlocking}  
 *println*("open(yes or not)")  
 var n = *readln*()  
 var open: Boolean  
 if (n == "yes") {  
 open = true  
 } else open = false  
 var r1: zad2 = zad2(login, pas, number, open)  
 reposit.add(r1)  
 }  
 } catch (e: Exception) {  
 *println*("error")  
 }  
 }  
 try{  
 var repos=reposit.*first*().SortedRep(reposit.*toList*())  
 for(i in 0..repos.*count*()-1)  
 {  
 *println*(repos[i].Get\_info())  
 }  
 }catch (e:Exception){  
  
 }  
  
 main1()  
**}**

**Тестовые ситуации**

****