Министерство образования и молодежной политики Свердловской области

ГАПОУ СО «Екатеринбургский колледж транспортного строительства»

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

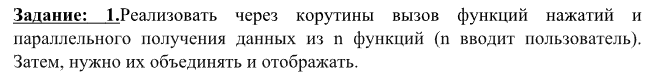
Выполнил: Рубцова Диана Алексеевна

Группа: ПР-21

Преподаватель: Мирошниченко Г.В

2024

**Задание:**



**Входные данные :**

s: Int-сколько раз будет выполняться функция

weight: Double - высотa равнобедренного треугольника

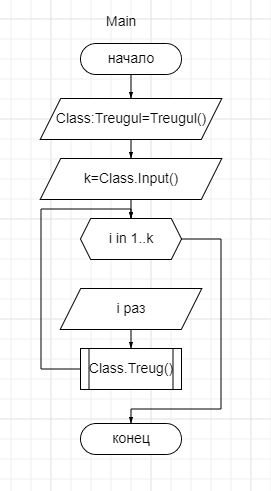
height: Double -основание равнобедренного треугольника

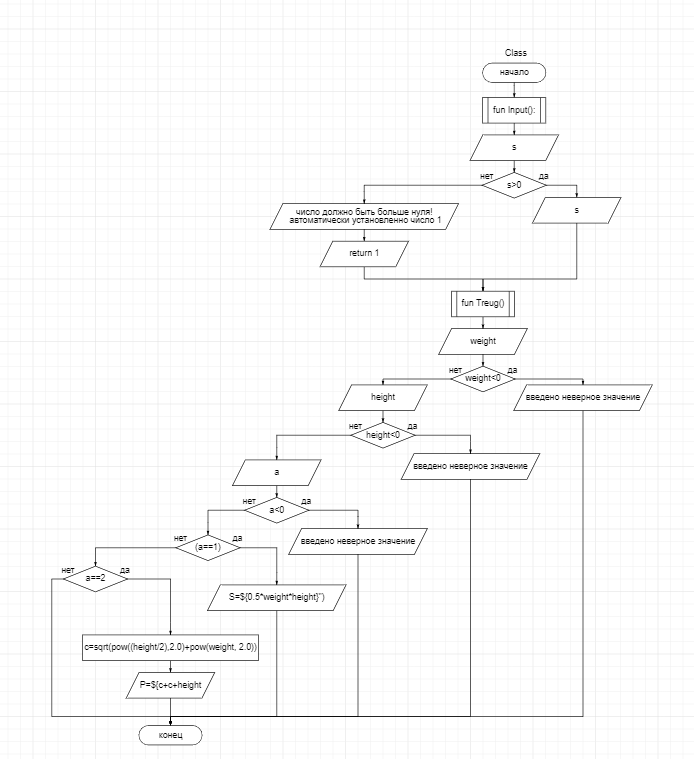
**Выходные данные:**

Double S-площадь треугольника

Double P-периметр треугольника

**Блок-схема**





**Листинг программы (если есть)**

**Main**

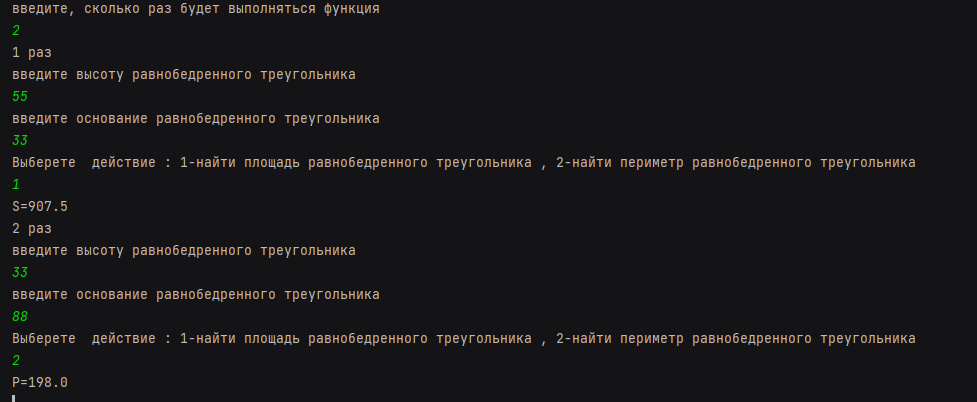
import kotlinx.coroutines.\*  
suspend fun main(){  
 var Class:Treugul=Treugul()  
 var k=Class.Input()  
 GlobalScope.launch**{** for(i in 1..k)  
 {  
 *println*("$i раз")  
 Class.Treug()  
 }  
 **}** *runBlocking* **{**delay( 30000L)**}**

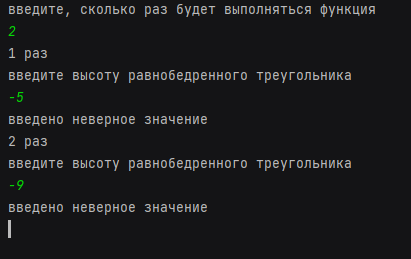
}

Class

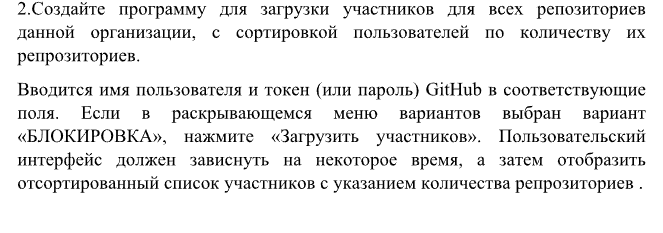
open class Treugul {  
 fun Input():Int  
 {  
 *println*("введите, сколько раз будет выполняться функция")  
 var s= *readLine*()!!.toInt()  
 if(s>0) return s  
 else *println*("число должно быть больше нуля! автоматически установленно число 1")  
 return 1  
 }  
 fun Treug(){  
 *println*("введите высоту равнобедренного треугольника") ;  
 var weight = *readLine*()!!.toDouble()  
 if (weight<0){ *println*("введено неверное значение")}  
 else {  
 *println*("введите основание равнобедренного треугольника");  
 var height = *readLine*()!!.toDouble()  
 if (height<0){ *println*("введено неверное значение")}  
 else {  
 *println*("Выберете действие : 1-найти площадь равнобедренного треугольника , 2-найти периметр равнобедренного треугольника")  
 var a= *readLine*()!!.toInt()  
 if (a<0){ *println*("введено неверное значение")}  
 var c=0.0  
 if(a==1)  
 *println*("S=${0.5\*weight\*height}")  
 else if (a==2){  
 c=*sqrt*(pow((height/2),2.0)+pow(weight, 2.0))  
 *println*("P=${c+c+height}")}  
 }  
 }  
 }  
}

**Тестовые ситуации**





**Задание:**



**Входные данные :**

login:String-логин пользователя

parol:String-пароль пользователя

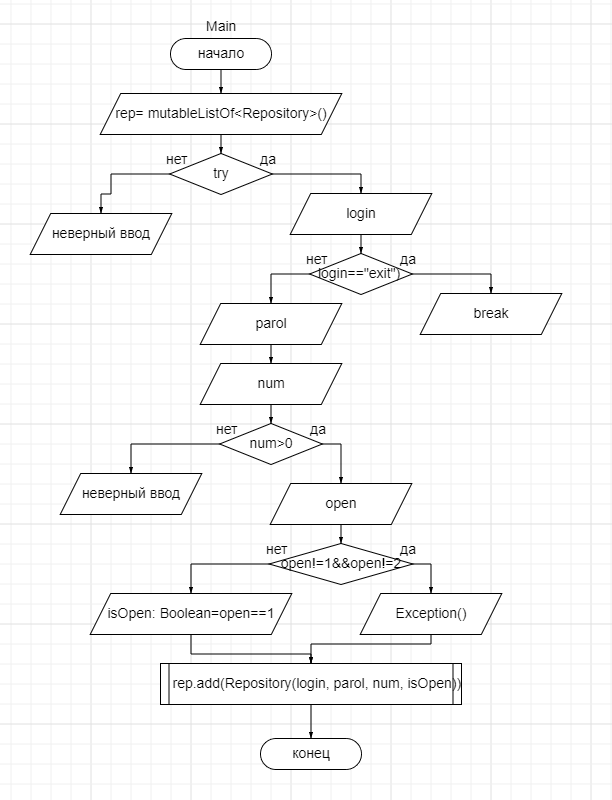
num:Int -кол-во репозиториев

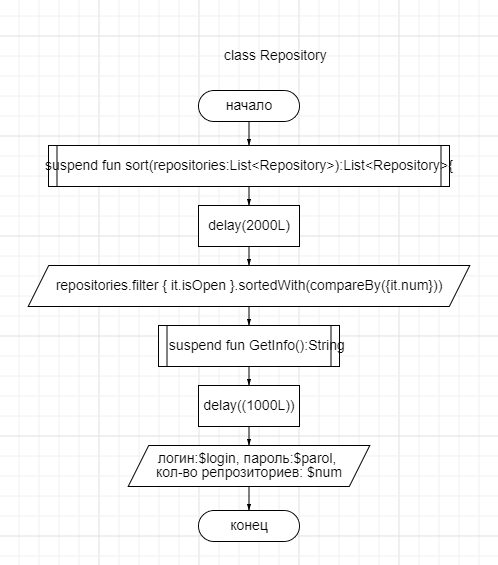
isOpen:Boolean-ответ на вопрос

**Выходные данные:**

Сообщение тип String с отсортированным списком пользователей и их данными

**Блок-схема**





**Листинг программы (если есть)**

**Main**

import kotlinx.coroutines.\*  
import kotlin.Exception  
import kotlin.system.exitProcess  
fun main()= *runBlocking***{** var rep= *mutableListOf*<Repository>()  
 while (true){  
 try{  
 *println*("\nЗагрузка нового участника")  
 *println*("Логин (exit для выхода)")  
 val login= *readLine*().*toString*()  
 if(login=="exit") break  
 *println*("Пароль")  
 val parol= *readLine*().*toString*()  
 *println*("Kол-во репозиториев")  
 val num= *readLine*()!!.*toInt*()  
 if(num>0) {  
 *println*("репозитории открыты?(1 - да, 2 - нет)")  
 val open= *readLine*()?.*toInt*()  
 if (open!=1&&open!=2) throw Exception()  
 val isOpen: Boolean=open==1  
 rep.add(Repository(login, parol, num, isOpen))  
 }  
 else  
 *println*("неверный ввод");  
  
 }  
 catch (e:Exception){ *println*("неверный ввод")}  
 }  
 val sort=rep.*first*().sort(rep)  
 sort.*forEach*(**{** *println*(**it**.GetInfo())**}**)  
**}**

**Class**

import kotlinx.coroutines.delay  
class Repository (var login:String, var parol:String, var num:Int, var isOpen:Boolean)  
{  
 suspend fun sort(repositories:List<Repository>):List<Repository>{  
 delay(2000L)  
 return repositories.*filter* **{ it**.isOpen **}**.*sortedWith*(*compareBy*(**{it**.num**}**))  
 }  
 suspend fun GetInfo():String{  
 delay((1000L))  
 return "логин:$login, пароль:$parol, кол-во репрозиториев: $num"  
 }  
}

**Тестовые ситуации**

