Министерство образования и молодежной политики Свердловской области



ГАПОУ СО «Екатеринбургский колледж транспортного строительства»

Отчёт по программе «**Практическое задание 12**»

Выполнил: Форрат Лия Константиновна

Группа: ПР-21

Преподаватель: Мирошниченко Г.В

2024

**Задание 1**

**Входные данные**

**Int save-**сколько раз будет выполняться функция

**Double kat1-**первый катет прямоугольного треугольника

**Double kat2-**второй катет прямоугольного треугольника

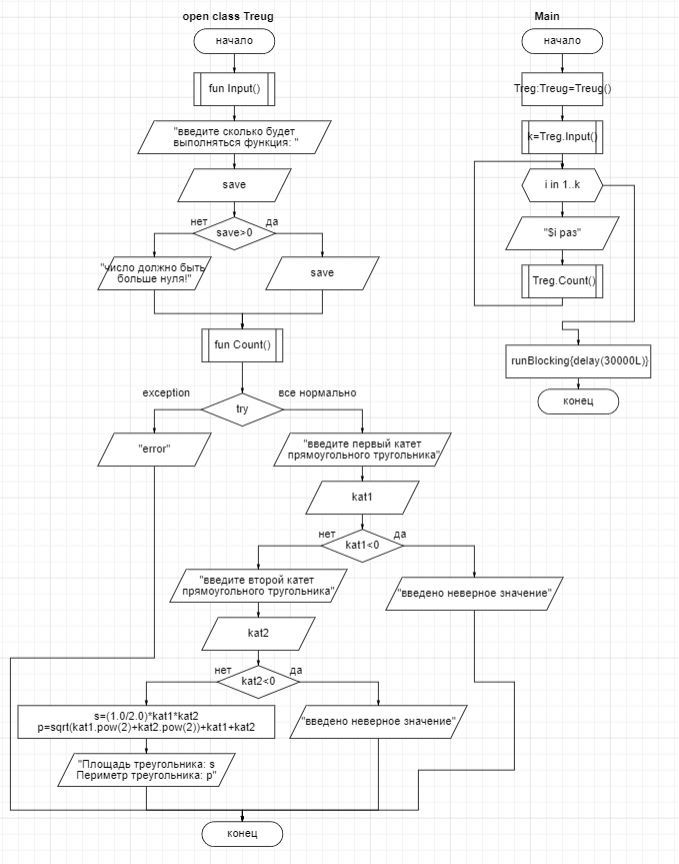
**Выходные данные**

Сообщения тип String

**Double s-п**лощадь треугольника

**Double p-**периметр треугольника

**Блок-схема**

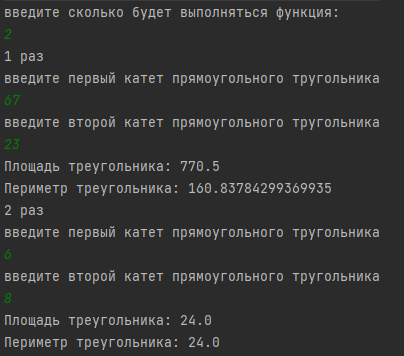


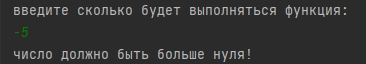
**Листинг программы (если есть)**

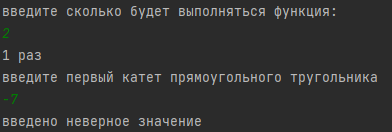
import kotlinx.coroutines.\*  
import kotlinx.coroutines.launch  
import kotlinx.coroutines.delay  
  
suspend fun main(){  
 var Treg:Treug=Treug()  
 var k=Treg.Input()  
 GlobalScope.*launch***{** for (i in 1..k){  
 *println*("$i раз")  
 Treg.Count()  
 }  
 **}** *runBlocking***{**delay(30000L)**}**}

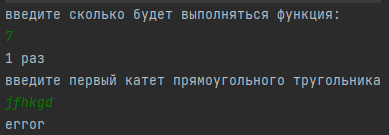
import kotlinx.coroutines.\*  
import kotlin.math.\*  
open class Treug () {  
 var kat1: Double = 1.0  
 var kat2: Double = 1.0  
 fun Input():Int  
 {  
 *println*("введите сколько будет выполняться функция: ")  
 var save=*readLine*()!!.*toInt*()  
 if(save>0)  
 return save  
 else *println*("число должно быть больше нуля!")  
 return 1  
 }  
 fun Count(){  
 try{  
 *println*("введите первый катет прямоугольного тругольника")  
 kat1 = *readLine*()!!.*toDouble*()  
 if(kat1<0){*println*("введено неверное значение")}  
 else{  
 *println*("введите второй катет прямоугольного тругольника")  
 kat2 = *readLine*()!!.*toDouble*()  
 if(kat2<0) {*println*("введено неверное значение") }  
 else {  
 var s=(1.0/2.0)\*kat1\*kat2  
 var p=*sqrt*(kat1.*pow*(2)+kat2.*pow*(2))+kat1+kat2  
 *println*("Площадь треугольника: $s\nПериметр треугольника: $p")  
 }  
 }  
 }catch(e:Exception){*println*("error")}}  
}

**Тестовые ситуации**









**Задание 2**

**Входные данные**

**String log-**логин пользователя

**String pas-**пароль пользователя

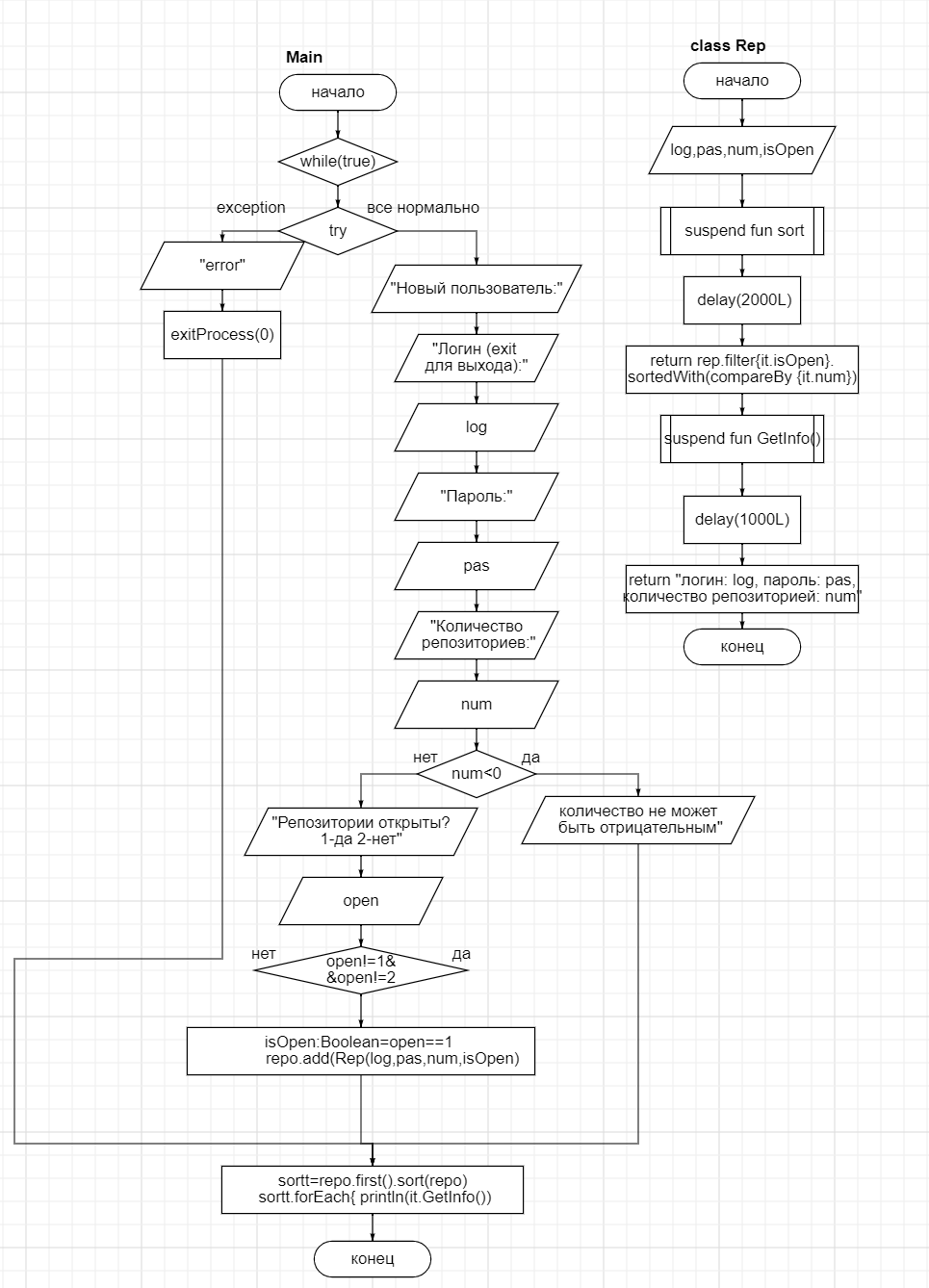
**Int num**-количество репозиториев

**Int open**-ответ на вопрос

**Выходные данные**

Сообщения тип **String** с отсортированным списком пользователей и их данными

**Блок-схема**

****

**Листинг программы (если есть)**

import kotlinx.coroutines.\*  
import kotlin.system.exitProcess  
fun main()=*runBlocking***{** var repo=*mutableListOf*<Rep>()  
 while(true){  
 try{  
 *println*("Новый пользователь:")  
 *println*("Логин (exit для выхода):")  
 val log=*readLine*().*toString*()  
 if(log=="exit") break  
 *println*("Пароль:")  
 val pas=*readLine*().*toString*()  
 *println*("Количество репозиториев:")  
 val num=*readLine*()!!.*toInt*()  
 if (num<0) *println*("количество не может быть отрицательным")  
 else {  
 *println*("Репозитории открыты?\n1-да\n2-нет")  
 val open=*readLine*()!!.*toInt*()  
 if(open!=1&&open!=2) throw Exception()  
 val isOpen:Boolean=open==1  
 repo.add(Rep(log,pas,num,isOpen))  
 }  
 }catch(e:Exception) {*println*("error")  
 *exitProcess*(0)}  
 }  
 val sortt=repo.*first*().sort(repo)  
  
 sortt.*forEach***{** *println*(**it**.GetInfo())**}  
}**

import kotlinx.coroutines.delay  
class Rep(var log:String,var pas:String, var num:Int, var isOpen: Boolean)  
{  
 suspend fun sort(rep: List<Rep>):List<Rep>{  
 delay(2000L)  
 return rep.*filter***{it**.isOpen**}**.*sortedWith*(*compareBy* **{it**.num**}**)  
 }  
 suspend fun GetInfo():String{  
 delay(1000L)  
 return "логин: $log, пароль: $pas, количество репозиторией: $num"  
 }  
  
}

**Тестовые ситуации**

