Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерного проектирования

Кафедра проектирования информационно-компьютерных систем

Отчёт

К лабораторной работе № 2

на тему:

**ЦИКЛЫ, ДИАПАЗОНЫ И МАССИВЫ**

Проверил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Усенко Ф. В.

(подпись)

Выполнил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.А. Грицков

(подпись) 214302

Минск 2024

**Вариант 6: Вывод чисел Фибоначчи с условиями.**

Условие: Напишите программу, которая выводит все числа Фибоначчи до заданного числа *N*, но только те, которые являются простыми. Программа должна использовать эффективный алгоритм проверки простоты чисел.

**Код программы:**

fun main() {  
 val n = 10  
 val fibNumberns = *getFibonacci*(n)  
 *println*("Числа Фибоначчи до $n -> $fibNumberns")  
 val maxFib = fibNumberns.*maxOrNull*() ?: 0  
 val primeSet = *sieveOfEratosthenes*(maxFib)  
  
 *println*("Простые числа Фибоначчи: ")  
 for (num in fibNumberns) {  
 if (num in primeSet) {  
 *print*(" $num")  
 }  
 }  
}  
  
fun getFibonacci(n: Int) : List<Int>{  
 val fibList = *mutableListOf*<Int>()  
 var a = 0  
 var b = 1  
 for (i in 0 .. n) {  
 fibList.add(a)  
 var c = a + b  
 a = b  
 b = c  
 }  
 return fibList  
}  
  
fun sieveOfEratosthenes(max: Int) : Set<Int> {  
 val isPrime = BooleanArray(max + 1) **{**true**}** isPrime[0] = false  
 isPrime[1] = false  
  
 for (i in 2..max) {  
 if (isPrime[i]) {  
 for (j in i\*2..max *step* i) {  
 isPrime[j] = false  
 }  
 }  
 }  
 return (2..max).*filter* **{** isPrime[**it**] **}**.*toSet*()  
}

**Контрольные вопросы:**

1. Какие типы циклов существуют в *Kotlin*, и как они используются?

– Цикл *for*: Используется для выполнения набора инструкций заданное количество раз или для итерации по элементам коллекции (например, массиву).

– Цикл *While*: Выполняет блок кода до тех пор, пока условие истинно. Если условие становится ложным, выполнение цикла прекращается.

– Цикл *do…While*: **:** Отличается от *while* тем, что сначала выполняется тело цикла, а затем проверяется условие.

1. Как работает цикл *for* для итерации по элементам массива? Приведите пример.

– Данный цикл последовательно перебирает все элементы массива и выполняет заданную инструкцию. Пример:

val numbers = arrayOf(1, 2, 3, 4, 5)

for (number in numbers) {

println(number)

}

1. Чем отличается цикл *while* от цикла *for*?

– Цикл *while* выполняется, пока не будет нарушено условие выполнения (*Boolean*). Цикл *for* выполняется, пока не будет выполнено арифметическое условие.

1. Какую роль выполняет цикл *do...while*, и в чем его отличие от *while*?

– Данный цикл выполняет ту же роль, что и цикл *while*, только, в отличие от while, цикл выполнится в любом случае хотя бы один раз.

1. Что такое диапазон в *Kotlin*, и как его создать?

– Диапазоном называется набор последовательных значений [*a,b*]. Диапазоны определяются с помощью оператора ... Диапазоны могут использоваться для итераций в циклах или для проверки принадлежности значению. Пример:

val x = 3

if (x in 1..5) {

println("$x is in the range")

}

1. Как использовать оператор *in* для проверки принадлежности значения диапазону?

Оператор *in* используется для проверки, находится ли значение в заданном диапазоне. Например, чтобы проверить, находится ли число 5 в диапазоне от 1 до 10*: if (5 in 1..10)*

1. Оператор !*in* используется для проверки, что значение не находится в заданном диапазоне. Например, чтобы проверить, что число 15 не находится в диапазоне от 1 до 10: *if (15 !in 1..10)*
2. Реверсированные диапазоны создаются с помощью метода *downTo*. Например, чтобы создать диапазон от 10 до 1*: for (i in 10 downTo 1)*
3. Шаг диапазона задается с помощью метода *step*. Например, чтобы создать диапазон от 1 до 10 с шагом 2: *for (i in 1..10 step 2)*
4. Массив чисел создается с помощью функции *arrayOf*. Например: *val* массив = *arrayOf*(1, 2, 3)
5. Обращение к элементу массива по индексу выполняется с помощью квадратных скобок. Например, чтобы получить первый элемент массива: массив[0]. Если индекс выйдет за пределы массива, будет выброшено исключение *ArrayIndexOutOfBoundsException*.
6. Значение элемента массива изменяется с помощью квадратных скобок. Например, чтобы изменить значение второго элемента массива: массив[1] = 42
7. Перебор элементов массива выполняется с помощью цикла *for*. Например: *for* (элемент *in* массив) { *println*(элемент) }
8. Массив фиксированного размера — это массив, размер которого задается при создании и не может быть изменен. Например, чтобы создать массив из 5 элементов: *val* массив = *Array*(5) { 0 }
9. Массив чисел сортируется с помощью метода *sort*. Например: массив.*sort*()
10. Для удаления элемента из массива нужно создать новый массив, не содержащий этот элемент. Например: *val* новыйМассив = массив.*filter* { *it* != 2 }.*toTypedArray*()
11. Метод *joinToString* создает строку из элементов массива с заданным разделителем. Метод *sort* сортирует массив. Например: массив.*joinToString*(", ") и массив.*sort*()
12. Два массива объединяются с помощью оператора +. Например: *val* объединенныйМассив = массив1 + массив2
13. Упорядоченность массива проверяется с помощью метода *zipWithNext*. Например: val упорядочен *= массив.zipWithNext().all { it.first <= it.second* }
14. Количество элементов, соответствующих условию, подсчитывается с помощью метода *count*. Например: *val* количество = массив.count { *it* % 2 == 0 }