

Лабораторная 5

Ермолаева Наталья ИС232

```
import kotlin.random.Random

import java.util.*

import java.time.LocalDate

import java.time.YearMonth

import java.time.format.DateTimeFormatter

fun main() {

    println("\n1")

    var i = 1

    while (i <= 10) {

        println(i)

        i++

    }

    println("\n2")

    var a = 2

    while (a <= 20) {

        println(a)

        a += 2

    }

    println("\n3")

    println("Введите число:")
```

```
var n = readln().toInt()
```

```
var r = 0
```

```
while (n > 0) {
```

```
    r += n
```

```
    n--
```

```
}
```

```
println(r)
```

```
println("\n4")
```

```
println("Введите число:")
```

```
var n1 = readln().toInt()
```

```
var r1 = 0
```

```
while (n > 0) {
```

```
    r1 *= n
```

```
    n1--
```

```
}
```

```
println("\n5")
```

```
println("Введите число для проверки на простоту:")
```

```
val number = readLine()?.toIntOrNull()
```

```
if (number != null && number > 1) {
```

```
    // Проверка, является ли число простым
```

```

        if (isPrime(number)) {

            println("Число $number является простым.")

        } else {

            println("Число $number не является простым.")

        }

    } else {

        println("Ошибка: Введите корректное число больше 1.")

    }

}

```

// Функция для проверки, является ли число простым

```

fun isPrime(n: Int): Boolean {

    // Проверка делителей от 2 до квадратного корня из n
    for (i in 2..Math.sqrt(n.toDouble()).toInt()) {

        if (n % i == 0) {

            return false // Если найден делитель, число не простое

        }

    }

    return true // Если делителей нет, число простое

```

```

println("\n6)")

```

```

var i1 = 1

```

```

while (i1 <= 10) {

```

```

    var i2 = 1

```

```

        while (i2 <= 10) {
            println(i1 * i2)
            i2++
        }
        i1++
    }

    println("\n7")
    print("Введите количество чисел Фибоначчи (N): ")
    val n2 = readLine()?.toIntOrNull()

    if (n2 != null && n2 > 0) {
        val fibonacciNumbers = generateFibonacci(n)
        println("Первые $n2 чисел Фибоначчи:
${fibonacciNumbers.joinToString(", ")}")
    } else {
        println("Ошибка: введите корректное положительное число.")
    }
}

fun generateFibonacci(n2: Int): List<Int> {
    val fibonacci = mutableListOf(0, 1)

    for (i in 2 until n2) {
        val nextNumber = fibonacci[i - 1] + fibonacci[i - 2]
    }
}

```

```

        fibonacci.add(nextNumber)
    }

    return fibonacci.take(n2)

    println("\n8)")
    print("Введите первое число: ")
    val number1 = readLine()?.toIntOrNull()

    print("Введите второе число: ")
    val number2 = readLine()?.toIntOrNull()

    if (number1 != null && number2 != null) {
        val gcd = findGCD(number1, number2)
        println("НОД чисел $number1 и $number2 равен $gcd")
    } else {
        println("Ошибка: введите корректные целые числа.")
    }
}

fun findGCD(a: Int, b: Int): Int {
    var num1 = a
    var num2 = b

```

```
while (num2 != 0) {  
    val temp = num2  
    num2 = num1 % num2  
    num1 = temp  
}
```

```
return num1
```

```
println("\n9)")
```

```
print("Введите строку: ")
```

```
val inputString = readLine()
```

```
if (!inputString.isNullOrEmpty()) {  
    val reversedString = inputString.reversed()  
    println("Строка в обратном порядке: $reversedString")  
} else {  
    println("Ошибка: введена пустая строка.")  
}
```

```
println("\n10)")
```

```
// Ввод числа от пользователя
```

```
println("Введите число:")
```

```
val number8 = readLine()?.toIntOrNull()
```

```
if (number8 != null && number8 >= 0) {
```

```
    // Вычисление суммы цифр
```

```
    var sum1 = 0
```

```
    var tempNumber = number8
```

```
    while (tempNumber != 0) {
```

```
        sum1 += tempNumber % 10 // Добавляем последнюю цифру к сумме
```

```
        tempNumber /= 10        // Убираем последнюю цифру из числа
```

```
    }
```

```
    // Вывод результата
```

```
    println("Сумма цифр числа $number8 равна $sum1")
```

```
} else {
```

```
    println("Ошибка: Введите корректное положительное число.")
```

```
}
```

```
println("\n11")
```

```
// Ввод двух строк от пользователя
```

```
println("Введите первую строку:")
```

```
val str1 = readLine()?.replace(" ", "").lowercase()
```

```
println("Введите вторую строку:")
```

```
val str2 = readLine()?.replace(" ", "").lowercase()

if (str1 != null && str2 != null) {

    // Проверка, являются ли строки анаграммами
    if (areAnagrams(str1, str2)) {

        println("Строки являются анаграммами.")
    } else {

        println("Строки не являются анаграммами.")
    }
} else {

    println("Ошибка: Введены некорректные данные.")
}
}

// Функция для проверки, являются ли строки анаграммами
fun areAnagrams(str1: String, str2: String): Boolean {

    // Если длины строк не равны, они не могут быть анаграммами
    if (str1.length != str2.length) {

        return false
    }

    // Преобразуем строки в списки символов и сортируем их
    val sortedStr1 = str1.toCharArray().sorted()
    val sortedStr2 = str2.toCharArray().sorted()
}
```



```
// Сравниваем отсортированные списки
```

```
return sortedStr1 == sortedStr2
```

```
println("\n12)")
```

```
println("Введите начальное число:")
```

```
val start = readLine()?.toIntOrNull()
```

```
// Ввод шага
```

```
println("Введите шаг:")
```

```
val step = readLine()?.toIntOrNull()
```

```
// Ввод количества элементов последовательности
```

```
println("Введите количество элементов последовательности:")
```

```
val count = readLine()?.toIntOrNull()
```

```
if (start != null && step != null && count != null && count > 0) {
```

```
    // Генерация последовательности
```

```
    val sequence = generateSequence(start) { it + step }.take(count)
```

```
    // Вывод последовательности
```

```
    println("Сгенерированная последовательность:")
```

```
    sequence.forEach { print("$it ") }
```

```

    } else {
        println("Ошибка: Введены некорректные данные.")
    }

    println("\n13")

    println("Число | Квадрат")
    println("-----")

    // Перебор чисел от 1 до 20
    for (i in 1..20) {
        val square = i * i
        println("$i\t| $square")
    }

    println("\n14")

    println("10 случайных чисел от 1 до 100:")

    for (i in 1..10) {
        val randomNumber = Random.nextInt(1, 101) // Генерация числа от 1 до
100

        println(randomNumber)
    }

    println("\n15")

    println("Введите строку для проверки на палиндром:")

    val input = readLine()

    if (input != null) {
        // Проверка, является ли строка палиндромом

```

```

        if (isPalindrome(input)) {

            println("Строка является палиндромом.")

        } else {

            println("Строка не является палиндромом.")

        }

    } else {

        println("Ошибка: Введена пустая строка.")

    }

}

```

// Функция для проверки, является ли строка палиндромом

```

fun isPalindrome(str: String): Boolean {

    // Убираем пробелы и приводим строку к нижнему регистру

    val cleanedStr = str.replace(" ", "").lowercase()

    // Сравниваем строку с её обратной версией

    return cleanedStr == cleanedStr.reversed()
}

```

```

println("\n16")

```

```

fun sumOfSquares(n: Int): Long {

    var sum = 0L

    for (i in 1..n) {

        sum += i.toLong() * i.toLong()

    }

    return sum

}

```

```
val n = 5
```

```
val result = sumOfSquares(n)
```

```
println("Сумма квадратов чисел от 1 до $n равна $result")
```

```
println("\n17")
```

```
val str = "Пример строки"
```

```
// Цикл for для перебора символов строки
```

```
for (char in str) {
```

```
    println(char)
```

```
}
```

```
println("\n18")
```

```
print("Введите высоту лестницы: ")
```

```
val height = readLine()!!.toInt()
```

```
for (i in 1..height) {
```

```
    println("#".repeat(i))
```

```
}
```

```
println("\n19")
```

```
val array = intArrayOf(23, 45, 12, 78, 34, 56)
```

```
// Вывод исходного массива
```

```
println("Исходный массив: ${array.contentToString()}")
```

```
// Сортировка пузырьком
```

```
for (i in 0 until array.size - 1) {
```

```
    for (j in 0 until array.size - i - 1) {
```

```
        if (array[j] > array[j + 1]) {
```

```
            // Меняем местами соседние элементы, если они не в порядке
```

```
            val temp = array[j]
```

```
            array[j] = array[j + 1]
```

```
            array[j + 1] = temp
```

```
        }
```

```
    }
```

```
}
```

```
// Вывод отсортированного массива
```

```
println("Отсортированный массив: ${array.contentToString()}")
```

```

println("\n20)")

print("Введите начальное значение диапазона: ")

val start = readLine()!!.toInt()

print("Введите конечное значение диапазона: ")

val end = readLine()!!.toInt()


println("Простые числа в диапазоне от $start до $end:")

for (num in start..end) {
    if (isPrime(num)) {
        println(num)
    }
}

}

fun isPrime(num: Int): Boolean {
    if (num <= 1) return false
    if (num <= 3) return true
    if (num % 2 == 0 || num % 3 == 0) return false
    var i = 5
    while (i * i <= num) {
        if (num % i == 0 || num % (i + 2) == 0) return false
        i += 6
    }
    return true
}

```

```
println("\n21")

print("Введите год: ")

val year = readLine()!!.toInt()

print("Введите месяц (1-12): ")

val month = readLine()!!.toInt()


val yearMonth = YearMonth.of(year, month)
val firstDayOfMonth = yearMonth.atDay(1)
val lastDayOfMonth = yearMonth.atEndOfMonth()


println("Все даты в месяце $month/$year:")
for (day in firstDayOfMonth..lastDayOfMonth) {
    println(day.format(DateTimeFormatter.ofPattern("dd/MM/yyyy")))
}
```

```
println("\n22")

val random = Random()

val secretNumber = random.nextInt(100) + 1

var guess: Int
```

```
var attempts = 0
```

```
println("Добро пожаловать в игру 'Угадай число'!")
```

```
println("Я загадал число от 1 до 100. Попробуйте его угадать.")
```

```
do {
```

```
    print("Введите ваше предположение: ")
```

```
    guess = readLine()!!.toInt()
```

```
    attempts++
```

```
    if (guess < secretNumber) {
```

```
        println("Ваше число меньше загаданного. Попробуйте еще раз.")
```

```
    } else if (guess > secretNumber) {
```

```
        println("Ваше число больше загаданного. Попробуйте еще раз.")
```

```
    } else {
```

```
        println("Поздравляю! Вы угадали число за $attempts попыток.")
```

```
    }
```

```
} while (guess != secretNumber)
```

```
println("\n23")
```

```
var continueOperation = true
```

```
var firstNumber: Int
```

```
var secondNumber: Int
```



```
var operation: String
```

```
while (continueOperation) {
```

```
    print("Введите первое число: ")
```

```
    firstNumber = readLine()!!.toInt()
```

```
    print("Введите второе число: ")
```

```
    secondNumber = readLine()!!.toInt()
```

```
    print("Выберите операцию (сложение или умножение): ")
```

```
    operation = readLine()!!.toLowerCase()
```

```
    when (operation) {
```

```
        "сложение" -> println("Результат сложения: ${firstNumber +  
secondNumber}")
```

```
        "умножение" -> println("Результат умножения: ${firstNumber *  
secondNumber}")
```

```
        "стоп" -> {
```

```
            println("Операция завершена.")
```

```
            continueOperation = false
```

```
        }
```

```
        else -> println("Неверная операция. Попробуйте снова.")
```

```
    }
```

```
}
```

```

println("\n24)")

val matrix = arrayOf(
    intArrayOf(1, 2, 3),
    intArrayOf(4, 5, 6),
    intArrayOf(7, 8, 9)
)

// Транспонируем матрицу
val transposedMatrix = transposeMatrix(matrix)

// Выводим результат
for (row in transposedMatrix) {
    println(row.toString(" "))
}

}

fun transposeMatrix(matrix: Array<IntArray>): Array<IntArray> {
    // Проверяем, что матрица не пустая
    if (matrix.isEmpty() || matrix[0].isEmpty()) {
        return emptyArray()
    }

    // Транспонируем матрицу
    val rows = matrix.size

```

```
val cols = matrix[0].size  
val transposed = Array(cols) { IntArray(rows) }
```

```
for (i in 0 until rows) {  
    for (j in 0 until cols) {  
        transposed[j][i] = matrix[i][j]  
    }  
}
```

```
return transposed
```

```
println("\n25")
```

```
for (i in 1..10) {  
    println("$i в кубе равно ${i * i * i}")  
}
```

```
println("\n26")
```

```
println("Введите число N:")
```

```
val n = readLine()?.toIntOrNull()
```

```
if (n != null && n > 0) {
```

```
    var sumEven = 0 // Сумма четных чисел
```

```
    var sumOdd = 0 // Сумма нечетных чисел
```

```

// Перебор чисел от 1 до N
for (i in 1..n) {
    if (i % 2 == 0) {
        sumEven += i // Если число четное, добавляем к сумме четных
    } else {
        sumOdd += i // Если число нечетное, добавляем к сумме
нечетных
    }
}

// Вывод результатов
println("Сумма четных чисел от 1 до $n: $sumEven")
println("Сумма нечетных чисел от 1 до $n: $sumOdd")
} else {
    println("Ошибка: Введите корректное положительное число.")
}

println("\n27")
print("Введите число N: ")
val n = readLine()!!.toInt()

// Выводим пирамиду

```

```

for (i in 1..n) {

    println((1..i).joinToString(" "))

}

println("\n28")

println("Введите количество чисел (N):")

val n5 = readLine()?.toIntOrNull()

if (n5 != null && n5 > 0) {

    val numbers = mutableListOf<Int>() // Список для хранения чисел

    // Ввод N чисел
    for (i in 1..n5) {

        println("Введите число #${i}:")

        val number5 = readLine()?.toIntOrNull()

        if (number5 != null) {

            numbers.add(number5) // Добавляем число в список

        } else {

            println("Ошибка: Введено некорректное число. Попробуйте
снова.")

            return

        }

    }

}

```

```
// Сортировка списка по возрастанию
numbers.sort()

// Вывод отсортированного списка
println("Числа в порядке возрастания:")
numbers.forEach { print("$it ") }
} else {
    println("Ошибка: Введите корректное положительное число N.")
}
```

```
println("\n29")
```

```
println("Введите число N:")
```

```
val n4 = readLine()?.toIntOrNull()
```

```
if (n4 != null && n4 > 0) {
```

```
    var sum2 = 0.0 // Переменная для хранения суммы ряда
```

```
    // Перебор чисел от 1 до N
```

```
    for (i in 1..n4) {
```

```
        sum2 += 1.0 / i // Добавляем к сумме 1/i
```

```
    }
```

```
    // Вывод результата
```

```
    println("Сумма ряда от 1 до 1/$n4: $sum2")
```

```
} else {  
    println("Ошибка: Введите корректное положительное число.")  
}  
  
println("\n30")  
  
println("Введите целое число:")  
  
val number6 = readLine()?.toIntOrNull()  
  
if (number6 != null) {  
    // Конвертация числа в двоичную систему  
  
    val binaryString = number6.toBinaryString()  
  
    println("Двоичное представление числа $number6: $binaryString")  
} else {  
    println("Ошибка: Введите корректное целое число.")  
}  
  
}  
  
// Функция для конвертации числа в двоичную строку  
  
fun Int.toBinaryString(): String {  
    return this.toString(2) // Используем встроенную функцию toString с  
    основанием 2  
}
```