Практическая №5

1.

```
fun main () {
    for (i in 1..10){
        println(i)
    }
}
```

2.

```
fun main () {
    for (i in 1..20) {
        if (i % 2 == 0) {
            println(i)
        }
    }
    for (i in 2..20 step 2) {
        println(i)
    }
}
```

3.

```
fun main(){
    print("Введите число N: ")
    val n = readLine()?.toIntOrNull() ?: return
    var sum = 0
    for (i in 1..n){
        sum += i
    }
    println("Сумма чисел от 1 до $n равна $sum")
}
```

4.

```
fun main(){
    print("Введите число: ")
    val number = readLine()?.toIntOrNull() ?: return
    var factorial= 1L
    for (i in 1..number){
        factorial *= i
    }
    println("Факториал числа $number равен $factorial")
}
```

5.

```
fun isPrime (number: Int): Boolean {
    if (number <= 1) return false
    if (number <= 3) return true
    if (number % 2 ==0 || number % 3 == 0) return false
    var i = 5
    while (i * i <= number) {
        if (number % i == 0 || number % (i+2) == 0) return false
        i += 6
    }
    return true
}
fun main() {
    print("Введите число: ")
    val number =
        readLine()?. toIntOrNull()?: return
    if (isPrime(number)) {
        println("$number - простое число.")
    } else {
```

```
        println("$number - составное число." )
    }
}
```

6.

```
fun main(){
    for (i in 1..10){
        for (j in 1..10){
            print("${i * j}\t")
        }
        println()
    }
}
```

7.

```
fun fibonacci(n: Int): List<Int> {
    val sequence = mutableListOf(1,1)
    for (i in 2 until n){
        sequence.add(sequence[i - 1] + sequence[i - 2])
    }
    return sequence.take(n).toList()
}
fun main() {
    print("введите количесво чисел Фибоначчи: ")
    val n = readLine()?.toIntOrNull() ?: return
    val fibSequence = fibonacci(n)
    println(fibSequence.joinToString(","))
}
```

8.

```
fun main() {
    println("Введите два числа:")
    val a = readLine()!!.toInt()
    val b = readLine()!!.toInt()
    val gcd = findGCD(a, b)
    println("Наибольший общий делитель ($a, $b) = $gcd")
}
fun findGCD(a: Int, b: Int): Int {
    var x = Math.abs(a)
    var y = Math.abs(b)
    while (y != 0) {
        val temp = y
        y = x % y
        x = temp
    }
    return x
}
```

9.

```
fun main() {
    print("Введите строку: ")
    val input = readLine().orEmpty()

    val reversed = input.reversed()
    println(reversed)
}
```

10.

```
fun digitSum(number: Int): Int {
    var num = number
    var sum = 0
    while (num > 0) {
        sum += num % 10
        num /= 10
    }
    return sum
}
fun main() {
    print("Введите число: ")
    val number = readLine()?.toIntOrNull() ?: return
    val sum = digitSum(number)
    println("Сумма цифр числа $number равна $sum")
}
```

11.

```
fun areAnagrams(s1: String, s2: String): Boolean {
    if (s1.length != s2.length) return false
    val charCounts = IntArray(256)
    for (c in s1) {
        charCounts[c.code]++
    }
    for (c in s2) {
        charCounts[c.code]--
    }
    for (count in charCounts) {
        if (count != 0) return false
    }
    return true
}
fun main() {
    print("Введите первую строку: ")
    val s1 = readLine()!!
    print("Введите вторую строку: ")
    val s2 = readLine()!!
    if (areAnagrams(s1, s2)) {
        println("Строки '$s1' и '$s2' являются анаграммами.")
    } else {
        println("Строки '$s1' и '$s2' не являются анаграммами.")
    }
}
```

12.

```
fun main() {
    print("Введите начальное число: ")
    val start = readLine()!!.toDouble()
    print("Введите шаг: ")
    val step = readLine()!!.toDouble()
    for (i in 0..9) {
        val value = start + i * step
        println(value)
    }
}
```

13.

```
fun main() {
    for (i in 1..20) {
        val square = i * i
        println("$i^2 = $square")
    }
}
```

14.

```
import kotlin.random.Random
fun main() {
    val randomNumbers = List(10) { Random.nextInt(1, 101) }
    println(randomNumbers)
}
```

15.

```
fun isPalindrome(s: String): Boolean {
    val cleanedS = s.filter { it.isLetterOrDigit() }.lowercase()
    return cleanedS == cleanedS.reversed()
}
fun main() {
    print("Введите строку: ")
    val input = readLine()!!
    if (isPalindrome(input)) {
        println("'$input' - палиндром.")
    } else {
        println("'$input' - не палиндром.")
    }
}
```

16.

```
fun sigmaSquares(N: Int): Long {
    var sum = 0L
    for (i in 1..N) {
        sum += (i * i).toLong()
    }
    return sum
}
fun main() {
    print("Введите число N: ")
    val N = readLine()!!.toInt()
    val result = sigmaSquares(N)
    println("Сумма квадратов чисел от 1 до $N равна $result")
}
```

17.

```
 fun main() {
    print("Введите строку: ")
    val inputString = readLine()!!
    for (char in inputString) {
        println(char)
    }
}
```

18.

```
fun main() {
    print("Введите высоту лестницы: ")
    val height = readLine()!!.toInt()
    for (i in 1..height) {
        println("#".repeat(i))
    }
}
```

19.

```
fun main() {
    print("Введите количество элементов массива: ")
    val size = readLine()!!.toInt()
    val array = Array(size) { readLine()!!.toInt() }
    for (i in 0 until size - 1) {
        for (j in i + 1 until size) {
            if (array[i] > array[j]) {
                val temp = array[i]
                array[i] = array[j]
                array[j] = temp
            }
        }
    }
    println("Отсортированный массив: ${array.joinToString(", ")}")
}
```

20.

```
fun isPrime(n: Int): Boolean {
    if (n <= 1) return false
    if (n == 2 || n == 3) return true
    if (n % 2 == 0 || n % 3 == 0) return false
    var i = 5
    while (i * i <= n) {
        if (n % i == 0 || n % (i + 2) == 0) return false
        i += 6
    }
    return true
}
fun main() {
    print("Введите начальное число диапазона: ")
    val start = readLine()!!.toInt()
    print("Введите конечное число диапазона: ")
    val end = readLine()!!.toInt()
    println("Простые числа в диапазоне от $start до $end:")
    for (num in start..end) {
        if (isPrime(num)) {
            println(num)
        }
    }
}
```

21.

```
import java.time.LocalDate
import java.time.YearMonth
import java.util.Scanner
fun main() {
    val scanner = Scanner(System.`in`)
    println("Введите год (например, 2025):")
    val year = scanner.nextInt()
    println("Введите месяц (1-12):")
    val month = scanner.nextInt()
    if (month in 1..12) {
        val yearMonth = YearMonth.of(year, month)
        val daysInMonth = yearMonth.lengthOfMonth()
        println("Даты в $month/$year:")
        for (day in 1..daysInMonth) {
            val date = LocalDate.of(year, month, day)
            println(date)
        }
    } else {
        println("Некорректный месяц. Пожалуйста, введите число от 1 до 12.")
```

```
    }
}
```

22.

```
import kotlin.random.Random
import java.util.Scanner
fun main() {
    val scanner = Scanner(System.`in`)
    val randomNumber = Random.nextInt(1, 101) // Случайное число от 1 до 100
    var guess: Int? = null
    var attempts = 0
    println("Угадайте число от 1 до 100!")
    while (guess != randomNumber) {
        println("Введите ваше предположение:")
        guess = scanner.nextInt()
        attempts++
        when {
            guess < randomNumber -> println("Слишком низко! Попробуйте еще
раз.")
            guess > randomNumber -> println("Слишком высоко! Попробуйте еще
раз.")
            else -> println("Поздравляем! Вы угадали число $randomNumber за
$attempts попыток.")
        }
    }
}
```

23.

```
import java.util.Scanner
fun main() {
    val scanner = Scanner(System.`in`)
    while (true) {
        println("Введите две цифры (или введите 'стоп' для выхода):")
        val input1 = scanner.nextLine()
        if (input1 == "стоп") break
        val input2 = scanner.nextLine()
        if (input2 == "стоп") break
        try {
            val number1 = input1.toDouble()
            val number2 = input2.toDouble()
            println("Введите операцию (сложение или умножение):")
            val operation = scanner.nextLine()
            when (operation) {
                "сложение" -> println("Результат сложения: ${number1 +
number2}")
                "умножение" -> println("Результат умножения: ${number1 *
number2}")
                else -> println("Неизвестная операция. Пожалуйста, введите
'сложение' или 'умножение'.")
            }
        } catch (e: NumberFormatException) {
            println("Пожалуйста, введите корректные цифры.")
        }
    }
    println("Программа завершена.")
}
```

24.

```
fun main() {
    val matrix = arrayOf(
        intArrayOf(1, 2, 3),
        intArrayOf(4, 5, 6),
        intArrayOf(7, 8, 9)
    )
    println("Исходная матрица:")
    printMatrix(matrix)
    val transposedMatrix = transposeMatrix(matrix)
    println("Транспонированная матрица:")
    printMatrix(transposedMatrix)
}
fun transposeMatrix(matrix: Array<IntArray>): Array<IntArray> {
    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
}
fun printMatrix(matrix: Array<IntArray>) {
    for (row in matrix) {
        println(row.joinToString(" "))
    }
}
```

25.

```
fun main() {
    println("Кубы чисел от 1 до 10:")
    for (i in 1..10) {
        val cube = i * i * i
        println("$i^3 = $cube")
    }
}
```

26.

```
fun main() {
    println("Введите число N:")
    val N = readLine()?.toIntOrNull()
    if (N != null && N > 0) {
        var sumEven = 0
        var sumOdd = 0
        for (i in 1..N) {
            if (i % 2 == 0) {
                sumEven += i
            } else {
                sumOdd += i
            }
        }
        println("Сумма четных чисел от 1 до $N: $sumEven")
        println("Сумма нечетных чисел от 1 до $N: $sumOdd")
    } else {
        println("Пожалуйста, введите положительное целое число.")
    }
}
```

27.

```
fun main() {
    println("Введите число N:")
    val N = readLine()?.toIntOrNull()
    if (N != null && N > 0) {
        for (i in 1..N) {
            for (j in 1..(N - i)) {
                print(" ")
            }
            for (k in 1..i) {
                print("$k ")
            }
            println()
        }
    } else {
        println("Пожалуйста, введите положительное целое число.")
    }
}
```

28.

```
fun main() {
    println("Введите количество чисел (N):")
    val N = readLine()?.toIntOrNull()
    if (N != null && N > 0) {
        val numbers = mutableListOf<Int>()
        println("Введите $N чисел:")
        for (i in 1..N) {
            val number = readLine()?.toIntOrNull()
            if (number != null) {
                numbers.add(number)
            } else {
                println("Некорректный ввод. Пожалуйста, введите целое
число.")
                return
            }
        }
        numbers.sort()
        println("Числа в порядке возрастания:")
        for (num in numbers) {
            print("$num ")
        }
    } else {
        println("Пожалуйста, введите положительное целое число.")
    }
}
```

29.

```
fun main() {
    println("Введите значение N:")
    val N = readLine()?.toIntOrNull()
    if (N != null && N > 0) {
        var sum = 0.0
        for (i in 1..N) {
            sum += 1.0 / i
        }
        println("Сумма ряда 1 + 1/2 + 1/3 + ... + 1/$N = $sum")
    } else {
        println("Пожалуйста, введите положительное целое число.")
    }
}
```

30.

```
fun main() {
    println("Введите целое число:")
    val number = readLine()?.toIntOrNull()
    if (number != null) {
        val binaryString = number.toString(2)
        println("Число $number в двоичной системе: $binaryString")
    } else {
        println("Пожалуйста, введите корректное целое число.")
    }
}
```