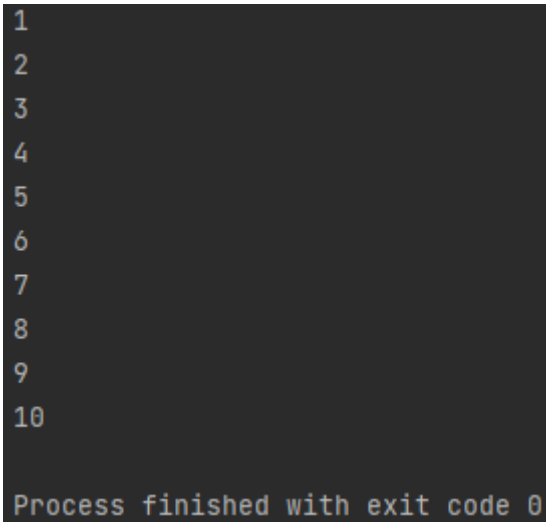


## Лаба №5

1. Вывод чисел от 1 до 10:

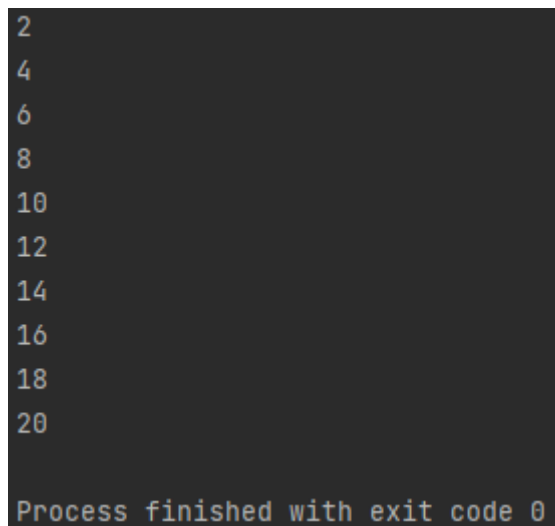
```
fun main() {  
    for (i in 1..10) {  
        println(i)  
    }  
}
```



```
1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
  
Process finished with exit code 0
```

2. Вывод четных чисел от 1 до 20: Напишите программу, которая выводит все четные числа от 1 до 20.

```
fun main() {  
    (1..20).filter { it % 2 == 0 }.forEach { println(it) }  
}
```



```
2  
4  
6  
8  
10  
12  
14  
16  
18  
20  
  
Process finished with exit code 0
```

3. Сумма чисел от 1 до N: Пользователь вводит число N, программа суммирует все числа от 1 до N и выводит результат.

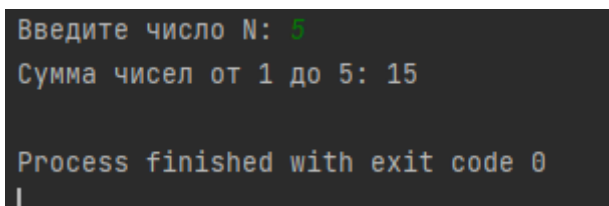
```

fun main() {
    print("Введите число N: ")
    val n = readLine()?.toIntOrNull() ?: 0 // Чтение числа N с обработкой ошибок

    val sum = (1..n).sum()
    println("Сумма чисел от 1 до $n: $sum")
}
fun main() {
    print("Введите число N: ")
    val n = readLine()?.toIntOrNull() ?: 0 // Чтение числа N с обработкой ошибок

    val sum = (1..n).sum()
    println("Сумма чисел от 1 до $n: $sum")
}

```



```

Введите число N: 5
Сумма чисел от 1 до 5: 15

Process finished with exit code 0

```

4. Факториал числа: Напишите программу, которая вычисляет факториал введенного пользователем числа.

```

fun main() {
    print("Введите неотрицательное целое число: ")
    val n = readLine()?.toIntOrNull() ?: return

    if (n < 0) {
        println("Факториал не определен для отрицательных чисел.")
        return
    }

    var factorial = 1L
    try {
        for (i in 1..n) {
            factorial = factorial * i
        }
    } catch (e: ArithmeticException) {
        println("Переполнение при вычислении факториала. Результат слишком большой.")
        return
    }

    println("Факториал $n равен $factorial")
}

```

```
Введите неотрицательное целое число: 10
Факториал 10 равен 3628800

Process finished with exit code 0
```

1. Проверка числа на простоту: Пользователь вводит число, программа определяет, является ли оно простым.

```
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
}

fun main() {
    print("Введите число: ")
    val num = readLine()?.toIntOrNull() ?: return
    if (isPrime(num)) println("$num является простым числом.") else println("$num не является простым числом.")
}
```

```
Введите число: 5
5 является простым числом.

Process finished with exit code 0
```

2. Вывод таблицы умножения: Напишите программу, которая выводит таблицу умножения от 1 до 10.

```

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

```

```

1  2  3  4  5  6  7  8  9  10
2  4  6  8  10 12 14 16 18 20
3  6  9  12 15 18 21 24 27 30
4  8  12 16 20 24 28 32 36 40
5  10 15 20 25 30 35 40 45 50
6  12 18 24 30 36 42 48 54 60
7  14 21 28 35 42 49 56 63 70
8  16 24 32 40 48 56 64 72 80
9  18 27 36 45 54 63 72 81 90
10 20 30 40 50 60 70 80 90 100

```

Process finished with exit code 0

7. Фибоначчи: Сгенерируйте первые N чисел Фибоначчи (где N вводит пользователь).

```

fun main() {
    print("Введите количество чисел Фибоначчи (N): ")
    val n = readLine()?.toIntOrNull() ?: return

    if (n <= 0) {
        println("Введите положительное число.")
        return
    }

    val fibonacciNumbers = mutableListOf(0L, 1L) // Список чисел Фибоначчи
    if (n > 2) {
        for (i in 2 until n) {
            fibonacciNumbers.add(fibonacciNumbers[i - 1] + fibonacciNumbers[i - 2])
        }
    }

    println(fibonacciNumbers.joinToString(", "))
}

```

```
Введите количество чисел Фибоначчи (N): 10
0, 1, 1, 2, 3, 5, 8, 13, 21, 34

Process finished with exit code 0
```

8. Наибольший общий делитель (НОД): Напишите программу, которая находит НОД двух введенных чисел с использованием алгоритма Евклида.

```
fun gcdIterativeImproved(a: Int, b: Int): Int {
    var a = Math.abs(a) // Преобразуем в абсолютное значение
    var b = Math.abs(b)
    while (b != 0) {
        val temp = b
        b = a % b
        a = temp
    }
    return a
}

fun main() {
    print("Введите первое число: ")
    val num1 = readLine()?.toIntOrNull() ?: return
    print("Введите второе число: ")
    val num2 = readLine()?.toIntOrNull() ?: return

    val result = gcdIterativeImproved(num1, num2)
    println("НОД ($num1, $num2) = $result")
}
```

```
Введите первое число: 7
Введите второе число: 40
НОД (7, 40) = 1

Process finished with exit code 0
```

9. Обратный порядок: Пользователь вводит строку, и программа выводит ее в обратном порядке.

```
fun main() {
    print("Введите строку: ")
    val inputString = readLine() ?: ""

    val sb = StringBuilder()
    for (i in inputString.length - 1 downTo 0) {
        sb.append(inputString[i])
    }
}
```

```

    }
    println("Строка в обратном порядке: ${sb.toString()}")
}

```

```

Введите строку: 123456789
Строка в обратном порядке: 987654321

Process finished with exit code 0

```

10. Сумма цифр числа: Напишите программу, которая находит сумму цифр введенного числа.

```

fun sumDigitsRecursive(number: Long): Long {
    return if (number == 0L) 0L else (number % 10) + sumDigitsRecursive(number / 10)
}

```

```

fun main() {
    print("Введите число: ")
    val number = readLine()?.toLongOrNull() ?: return

    println("Сумма цифр: ${sumDigitsRecursive(number)}")
}

```

```

Введите число: 72
Сумма цифр: 9

Process finished with exit code 0

```

11. Анаграммы: Программа проверяет, являются ли две введенные строки анаграммами.

```

fun areAnagramsMap(str1: String, str2: String): Boolean {
    val charCount1 = mutableMapOf<Char, Int>()
    val charCount2 = mutableMapOf<Char, Int>()

    for (char in str1.lowercase()) {
        charCount1[char] = charCount1.getOrDefault(char, 0) + 1
    }
    for (char in str2.lowercase()) {
        charCount2[char] = charCount2.getOrDefault(char, 0) + 1
    }
    return charCount1 == charCount2
}

```

```

fun main() {
    print("Введите первую строку: ")
}

```

```

val string1 = readLine() ?: ""
print("Введите вторую строку: ")
val string2 = readLine() ?: ""

if (areAnagramsMap(string1, string2)) {
    println("Строки являются анаграммами.")
} else {
    println("Строки не являются анаграммами.")
}
}

```

```

Введите первую строку: 32
Введите вторую строку: 23
Строки являются анаграммами.

Process finished with exit code 0

```

12. Числовая последовательность: Пользователь вводит начальное число и шаг, программа генерирует числовую последовательность.

```

fun main() {
    print("Введите начальное число: ")
    val start = readLine()?.toIntOrNull() ?: return
    print("Введите шаг: ")
    val step = readLine()?.toIntOrNull() ?: return
    print("Введите предел: ")
    val limit = readLine()?.toIntOrNull() ?: return

    print("Последовательность: ")
    var current = start
    while (current <= limit) {
        print("$current ")
        current += step
    }
    println()
}

```

```

Введите начальное число: 5
Введите шаг: 1
Введите предел: 10
Последовательность: 5 6 7 8 9 10

Process finished with exit code 0

```

13. Таблица квадратов: Выведите таблицу квадратов чисел от 1 до 20.

```
fun main() {
    println("Таблица квадратов:")
    (1..20).map { "$it * $it = ${it * it}" }.forEach(::println)
}
```

Таблица квадратов:

1	*	1	=	1
2	*	2	=	4
3	*	3	=	9
4	*	4	=	16
5	*	5	=	25
6	*	6	=	36
7	*	7	=	49
8	*	8	=	64
9	*	9	=	81
10	*	10	=	100
11	*	11	=	121
12	*	12	=	144
13	*	13	=	169
14	*	14	=	196
15	*	15	=	225
16	*	16	=	256
17	*	17	=	289
18	*	18	=	324
19	*	19	=	361
20	*	20	=	400

14. Генерация случайных чисел: Сгенерируйте и выведите 10 случайных чисел от 1 до 100.

```
import kotlin.random.Random
```

[illegible]



```
10 случайных чисел от 1 до 100:  
29  
37  
99  
43  
22  
12  
46  
80  
74  
93
```

15. Проверка палиндрома: Пользователь вводит строку, и программа проверяет, является ли она палиндромом.

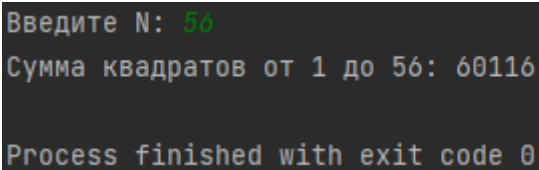
```
fun isPalindrome(str: String): Boolean {  
    val cleanStr = str.lowercase().replace(Regex("[^a-zA-Z0-9]"), "") //Удаление не буквенно-  
цифровых символов  
    val n = cleanStr.length  
    for (i in 0 until n / 2) {  
        if (cleanStr[i] != cleanStr[n - 1 - i]) {  
            return false  
        }  
    }  
    return true  
}  
  
fun main() {  
    print("Введите строку: ")  
    val inputString = readLine() ?: ""  
    if (isPalindrome(inputString)) {  
        println("Строка является палиндромом.")  
    } else {  
        println("Строка не является палиндромом.")  
    }  
}
```

```
Введите строку: 33  
Строка является палиндромом.  
  
Process finished with exit code 0  
|
```

16. Сигма (сумма квадратов): Найдите сумму квадратов всех чисел от 1 до N.

```
fun sumOfSquaresFunctional(n: Int): Long = (1..n).sumOf { (it * it).toLong() }
```

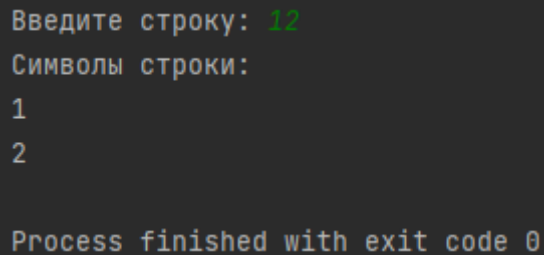
```
fun main() {  
    print("Введите N: ")  
    val n = readLine()?.toIntOrNull() ?: return  
  
    val sum = sumOfSquaresFunctional(n)  
    println("Сумма квадратов от 1 до $n: $sum")  
}
```



Введите N: 56  
Сумма квадратов от 1 до 56: 60116  
Process finished with exit code 0

17. Вывод символов: Напишите программу, которая выводит символы строки по одному, используя циклы.

```
fun main() {  
    print("Введите строку: ")  
    val str = readLine() ?: ""  
    var i = 0  
  
    println("Символы строки:")  
    while (i < str.length) {  
        println(str[i])  
        i++  
    }  
}
```



Введите строку: 12  
Символы строки:  
1  
2  
Process finished with exit code 0

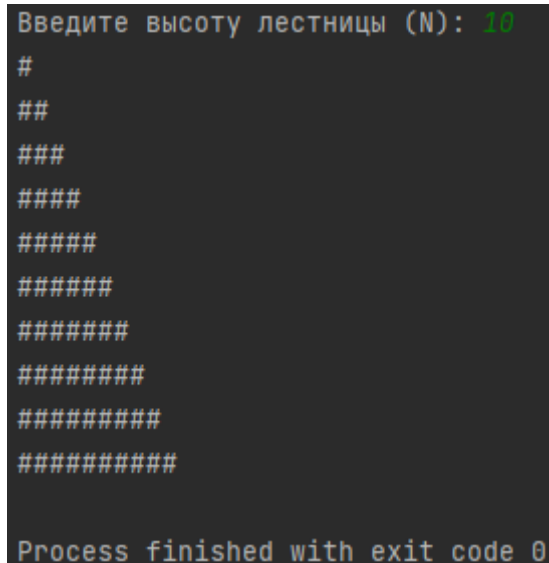
18. Задача на лестницу: Напишите программу, которая выводит лестницу из символа "#" высотой N, где N задает пользователь.

```

fun main() {
    print("Введите высоту лестницы (N): ")
    val n = readLine()?.toIntOrNull() ?: return

    for (i in 1..n) {
        println("#".repeat(i))
    }
}

```



```

Введите высоту лестницы (N): 10
#
##
###
####
#####
#####
#####
#####
#####
#####
#####
#####
Process finished with exit code 0

```

19. Сортировка списка: Используя цикл, напишите простую сортировку для двухзначных чисел в массиве.

```

fun main() {
    val numbers = arrayOf(34, 12, 45, 23, 56, 78, 11, 99, 21, 67)

    println("Исходный массив: ${numbers.joinToString(", ")}")

    // Сортировка пузырьком
    for (i in numbers.indices) {
        for (j in 0 until numbers.size - 1 - i) {
            if (numbers[j] > numbers[j + 1]) {
                // Меняем местами
                val temp = numbers[j]
                numbers[j] = numbers[j + 1]
                numbers[j + 1] = temp
            }
        }
    }

    println("Отсортированный массив: ${numbers.joinToString(", ")}")
}

```

```
}
```

```
Исходный массив: 34, 12, 45, 23, 56, 78, 11, 99, 21, 67
Отсортированный массив: 11, 12, 21, 23, 34, 45, 56, 67, 78, 99

Process finished with exit code 0
```

20. Простые числа в диапазоне: Выведите все простые числа в заданном пользователем диапазоне.

```
fun main() {
    println("Введите начало диапазона:")
    val start = readLine()!!.toInt()

    println("Введите конец диапазона:")
    val end = readLine()!!.toInt()

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

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

fun isPrime(num: Int): Boolean {
    if (num < 2) return false
    for (i in 2 until num) {
        if (num % i == 0) {
            return false
        }
    }
    return true
}
```

```
Введите начало диапазона:
4
Введите конец диапазона:
44
Простые числа в диапазоне от 4 до 44:
5
7
11
13
17
19
23
29
31
37
41
43

Process finished with exit code 0
```

21. Вывод даты: Пользователь вводит год и месяц, программа выводит все даты в этом месяце.

```
import java.time.LocalDate
import java.time.Month
```

```
fun main() {
    println("Введите год:")
    val year = readLine()!!.toInt()
    println("Введите месяц:")
    val month = readLine()!!.toInt()

    val daysInMonth = Month.of(month).length(LocalDate.of(year, month, 1).isLeapYear)
    for (day in 1..daysInMonth) {
        println("$year-$month-$day")
    }
}
```

```
} import java.time.LocalDate
import java.time.Month
```

```
fun main() {
    println("Введите год:")
    val year = readLine()!!.toInt()
    println("Введите месяц:")
    val month = readLine()!!.toInt()
```

```
val daysInMonth = Month.of(month).length(LocalDate.of(year, month, 1).isLeapYear)
for (day in 1..daysInMonth) {
    println("$year-$month-$day")
}
}
```

Введите год:

1987

Введите месяц:

7

1987-7-1

1987-7-2

1987-7-3

1987-7-4

1987-7-5

1987-7-6

1987-7-7

1987-7-8

1987-7-9

1987-7-10

1987-7-11

1987-7-12

1987-7-13

1987-7-14

1987-7-15

1987-7-16

1987-7-17

1987-7-18

1987-7-19

1987-7-20

1987-7-21

1987-7-22

1987-7-23

1987-7-24

1987-7-25

1987-7-26

1987-7-27

1987-7-28

1987-7-29

1987-7-30

1987-7-31

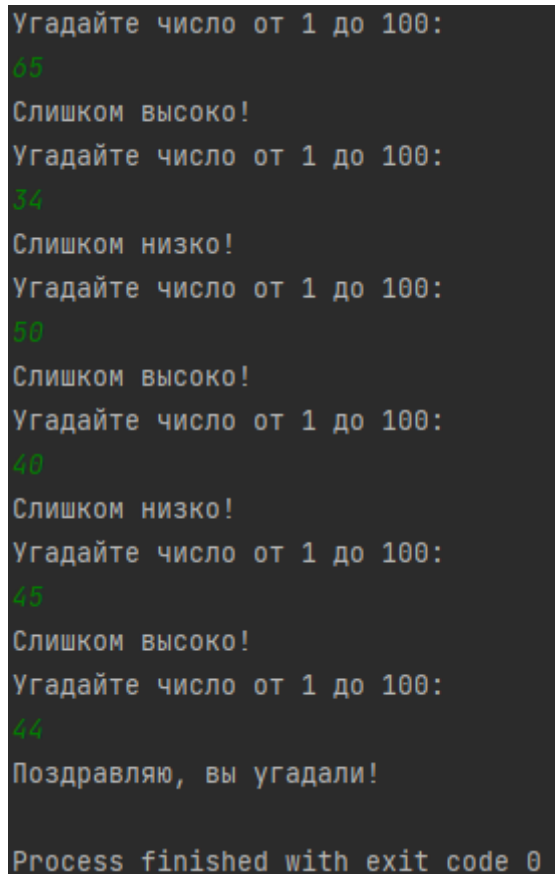
Process finished with exit code

22. Угадай число: Напишите игру, в которой пользователь должен угадать.

```
import kotlin.random.Random

fun main() {
    val numberToGuess = Random.nextInt(1, 101)
    var guess: Int? = null

    while (guess != numberToGuess) {
        println("Угадайте число от 1 до 100:")
        guess = readLine()!!.toInt()
        when {
            guess < numberToGuess -> println("Слишком низко!")
            guess > numberToGuess -> println("Слишком высоко!")
        }
    }
    println("Поздравляю, вы угадали!")
}
```



```
Угадайте число от 1 до 100:
65
Слишком высоко!
Угадайте число от 1 до 100:
34
Слишком низко!
Угадайте число от 1 до 100:
50
Слишком высоко!
Угадайте число от 1 до 100:
40
Слишком низко!
Угадайте число от 1 до 100:
45
Слишком высоко!
Угадайте число от 1 до 100:
44
Поздравляю, вы угадали!
Process finished with exit code 0
```

23. Сложение и умножение: Напишите программу, которая запрашивает у пользователя две цифры и повторяет сложение или умножение, до тех пор, пока пользователь не введет "стоп".

```
fun main() {
    while (true) {
```

```

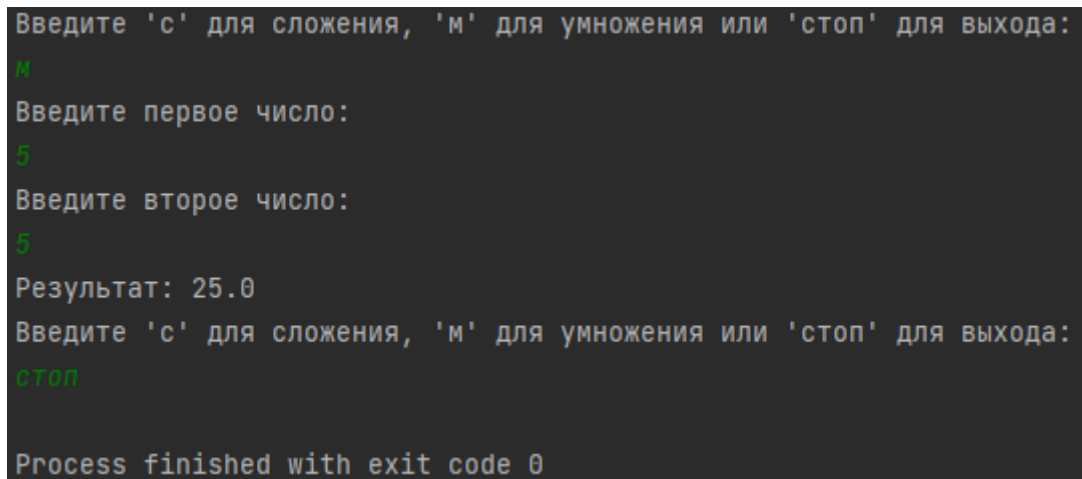
println("Введите 'с' для сложения, 'м' для умножения или 'стоп' для выхода:")
val operation = readLine()!!

if (operation == "стоп") break

println("Введите первое число:")
val a = readLine()!!.toDouble()
println("Введите второе число:")
val b = readLine()!!.toDouble()

when (operation) {
    "с" -> println("Результат: ${a + b}")
    "м" -> println("Результат: ${a * b}")
}
}
}

```



```

Введите 'с' для сложения, 'м' для умножения или 'стоп' для выхода:
м
Введите первое число:
5
Введите второе число:
5
Результат: 25.0
Введите 'с' для сложения, 'м' для умножения или 'стоп' для выхода:
стоп

Process finished with exit code 0

```

24. Транспонирование матрицы: Напишите программу, которая транспонирует матрицу (двумерный массив).

```

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

    val transposed = Array(matrix[0].size) { IntArray(matrix.size) }
    for (i in matrix.indices) {
        for (j in matrix[i].indices) {
            transposed[j][i] = matrix[i][j]
        }
    }

    for (row in transposed) {

```



```

        println(row.joinToString(" "))
    }
}

```

```

1 4 7
2 5 8
3 6 9

Process finished with exit code 0

```

25. Кубы чисел: Выведите кубы чисел от 1 до 10.

```

fun main() {
    for (i in 1..10) {
        println("$i^3 = ${i * i * i}")
    }
}

```

```

1^3 = 1
2^3 = 8
3^3 = 27
4^3 = 64
5^3 = 125
6^3 = 216
7^3 = 343
8^3 = 512
9^3 = 729
10^3 = 1000

Process finished with exit code 0

```

26. Сумма четных и нечетных чисел: Пользователь вводит N, программа считает сумму четных и нечетных чисел от 1 до N.

```

fun main() {
    println("Введите N:")
    val N = readLine()!!.toInt()
    val evenSum = (1..N).filter { it % 2 == 0 }.sum()
    val oddSum = (1..N).filter { it % 2 != 0 }.sum()
    println("Сумма четных: $evenSum, Сумма нечетных: $oddSum")
}

```

```
Введите N:
5
Сумма четных: 6, Сумма нечетных: 9
Process finished with exit code 0
```

27. Печать числа "пирамида": Напишите программу, которая выводит "пирамиду" из чисел от 1 до N.

```
fun main() {
    println("Введите N:")
    val N = readLine()!!.toInt()
    for (i in 1..N) {
        println((1..i).joinToString(" "))
    }
}
```

```
Введите N:
5
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
Process finished with exit code 0
```

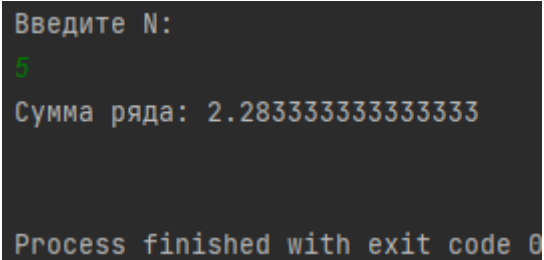
28. Определение порядка: Программа принимает N чисел и выводит их в порядке возрастания.

```
fun main() {
    println("Введите числа через пробел:")
    val numbers = readLine()!!.split(" ").map { it.toInt() }
    val sortedNumbers = numbers.sorted()
    println("Отсортированные числа: $sortedNumbers")
}
```

```
Введите числа через пробел:
11 43 47 8
Отсортированные числа: [8, 11, 43, 47]
Process finished with exit code 0
```

29. Сумма ряда: Напишите программу, которая находит сумму ряда  $1, 1/2, 1/3, \dots, 1/N$ .

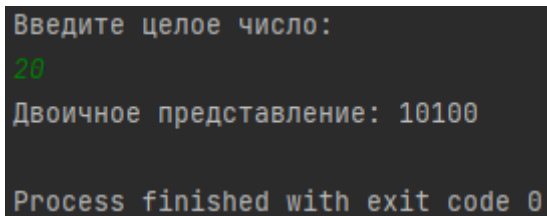
```
fun main() {  
    println("Введите N:")  
    val N = readLine()!!.toInt()  
    val seriesSum = (1..N).sumOf { 1.0 / it }  
    println("Сумма ряда: $seriesSum")  
}
```



```
Введите N:  
5  
Сумма ряда: 2.283333333333333  
  
Process finished with exit code 0
```

30. Конвертация в двоичную систему: Напишите программу, которая конвертирует целое число в двоичную систему.

```
fun main() {  
    println("Введите целое число:")  
    val number = readLine()!!.toInt()  
    val binary = Integer.toBinaryString(number)  
    println("Двоичное представление: $binary")  
}
```



```
Введите целое число:  
20  
Двоичное представление: 10100  
  
Process finished with exit code 0
```