

1.

```
fun main() {  
    val numbers = listOf(1, 2, 3, 4, 5)  
    println(sumOfList(numbers))  
}
```

```
fun sumOfList(list: List<Int>): Int {  
    return list.sum()  
}
```



15

Process finished with exit code 0

2.

```
fun maxMinDifference(numbers: List<Int>): Int {  
    if (numbers.isEmpty()) return 0  
    return numbers.maxOrNull()!! - numbers.minOrNull()!!  
}
```

```
fun main() {  
    val numbers = listOf(5, 2, 9, 1, 7, 6, 3)  
    println(maxMinDifference(numbers))  
}
```



8

Process finished with exit code 0

3.

```
fun main() {  
    val list1 = listOf(1, 2, 3)  
    val list2 = listOf(4, 5, 6)
```

```
println(mergeLists(list1, list2))  
}
```

```
fun mergeLists(list1: List<Int>, list2: List<Int>): List<Int> {  
    val result = mutableListOf<Int>()  
    result.addAll(list1)  
    result.addAll(list2)  
    return result  
}
```

4.

```
fun evaluateBet(prob: Double, prize: Double, pay: Double): Boolean {  
    return prob * prize > pay  
}
```

```
fun main() {  
    val prob = 0.5 // Пример вероятности  
    val prize = 100.0 // Пример приза  
    val pay = 40.0 // Пример ставки  
  
    val result = evaluateBet(prob, prize, pay)  
    println(result) // Вывод результата  
}
```

```
true
```

```
Process finished with exit code 0
```

6.

```
fun isSumLessThan100(num1: Int, num2: Int): Boolean {  
    return (num1 + num2) < 100  
}
```

```
fun main() {  
    val number1 = 30  
    val number2 = 50  
  
    val result = isSumLessThan100(number1, number2)  
    println(result)  
}
```

```
true  
  
Process finished with exit code 0
```

7.

```
fun isDivisibleBy100(number: Int): Boolean {  
    return number % 100 == 0  
}
```

```
fun main() {  
    val num = 250 // Пример числа  
  
    val result = isDivisibleBy100(num)  
    println(result) // Вывод результата  
}
```

```
false  
  
Process finished with exit code 0
```

8.

```

fun calculateFrames(minutes: Int, fps: Int): Int {
    val totalSeconds = minutes * 60
    return totalSeconds * fps
}

fun main() {
    val minutes = 2
    val fps = 144

    val totalFrames = calculateFrames(minutes, fps)
    println(totalFrames)
}

```

```

17280

Process finished with exit code 0

```

9.

```

fun isKPowerEqualToN(n: Int, k: Int): Boolean {
    return Math.pow(k.toDouble(), k.toDouble()) == n.toDouble()
}

fun main() {
    val n = 81 // Пример значения n
    val k = 3  // Пример значения k

    val result = isKPowerEqualToN(n, k)
    println(result) // Вывод результата
}

```

```

false

Process finished with exit code 0

```

10.

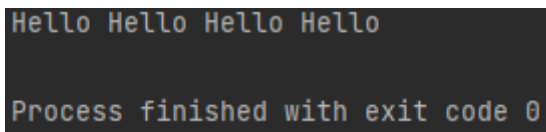
```

fun repeatString(txt: String, n: Int): String {
    return if (n <= 0) "" // Базовый случай: если n меньше или равно 0, возвращаем пустую строку
    else txt + repeatString(txt, n - 1) // Рекурсивный вызов с уменьшенным n
}

fun main() {
    val text = "Hello " // Пример строки
    val repetitions = 4 // Пример количества повторений

    val result = repeatString(text, repetitions)
    println(result) // Вывод результата
}

```



```

Hello Hello Hello Hello
Process finished with exit code 0

```

11.

```

fun evaluateEquation(equation: String): Double {
    val parts = equation.split(" ")
    var result = parts[0].toDouble()

    for (i in 1 until parts.size step 2) {
        val operator = parts[i]
        val nextValue = parts[i + 1].toDouble()

        result = when (operator) {
            "+" -> result + nextValue
            "-" -> result - nextValue
            "*" -> result * nextValue
            "/" -> result / nextValue
            else -> throw IllegalArgumentException("Unsupported operator: $operator")
        }
    }

    return result
}

```

```
}
```

```
fun main() {  
    val equation = "1 + 2 * 3 - 4 / 2" // Пример уравнения  
  
    try {  
        val result = evaluateEquation(equation)  
        println(result) // Вывод результата  
    } catch (e: Exception) {  
        println("Ошибка в уравнении: ${e.message}")  
    }  
}
```

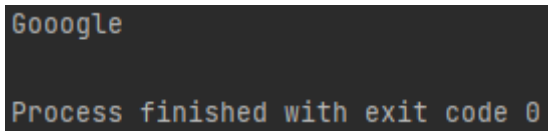
A screenshot of a terminal window with a dark background. The text '2.5' is displayed in a light blue or cyan color, indicating the output of the program.

Process finished with exit code 0

12.

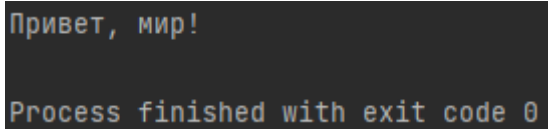
```
fun generateGoogleWord(number: Int): String {  
    // Проверяем, чтобы количество o было неотрицательным  
    if (number < 0) {  
        throw IllegalArgumentException("Количество букв o не может быть отрицательным")  
    }  
  
    // Формируем строку "G" + "o" * number + "gle"  
    return "G" + "o".repeat(number) + "gle"  
}  
  
fun main() {  
    val number = 3 // Пример числа  
  
    val result = generateGoogleWord(number)
```

```
println(result) // Вывод результата
}
```

A terminal window with a dark background. The first line shows the word "Gooogle" in a light blue font. The second line shows "Process finished with exit code 0" in a light green font.

13.

```
fun main() {
    val text = "Привет, мир!"
    println(text)
}
```

A terminal window with a dark background. The first line shows the Russian text "Привет, мир!" in a light blue font. The second line shows "Process finished with exit code 0" in a light green font.

14.

```
import java.util.Scanner
```

```
fun sum(a: Int, b: Int): Int {
    return a + b
}
```

```
fun main() {
    val scanner = Scanner(System.`in`) // Создаем объект Scanner для чтения ввода

    println("Введите первое число:")
    val number1 = scanner.nextInt() // Читаем первое число

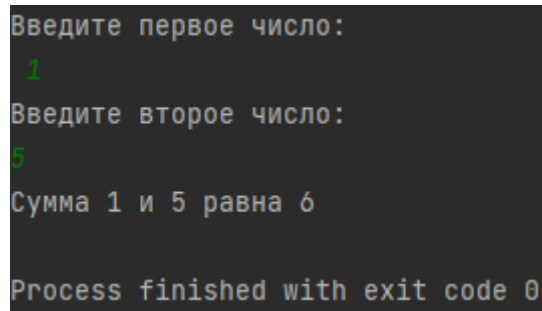
    println("Введите второе число:")
```

```

val number2 = scanner.nextInt() // Читаем второе число

val result = sum(number1, number2) // Вызываем функцию для вычисления суммы
println("Сумма $number1 и $number2 равна $result") // Вывод результата
}

```



```

Введите первое число:
1
Введите второе число:
5
Сумма 1 и 5 равна 6
Process finished with exit code 0

```

15.

```

fun isEven(number: Int): Boolean {
    return number % 2 == 0 // Если число делится на 2 без остатка, оно четное
}

fun main() {
    println("Введите число:")
    val input = readLine()?.toIntOrNull() // Читаем ввод и преобразуем в Int

    if (input != null) {
        val result = isEven(input) // Вызываем функцию для определения четности
        if (result) {
            println("$input - четное число")
        } else {
            println("$input - нечетное число")
        }
    } else {
        println("Ошибка: введите корректное число.")
    }
}

```



```
}
```

```
Введите число:  
0  
0 - четное число  
  
Process finished with exit code 0
```

16.

```
fun main() {  
    println(isEven(4)) // true  
    println(isEven(7)) // false  
    println(isEven(0)) // true  
}
```

```
fun isEven(number: Int): Boolean {  
    return number % 2 == 0  
}
```

```
true  
false  
true  
  
Process finished with exit code 0
```

17.

```
fun main() {  
    println("Введите число для вычисления факториала:")  
    val number = readLine()?.toIntOrNull()
```

```

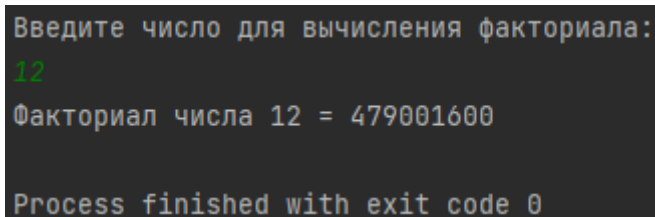
if (number != null && number >= 0) {
    println("Факториал числа $number = ${factorial(number)}")
} else {
    println("Пожалуйста, введите корректное положительное число")
}
}

```

```

fun factorial(n: Int): Long {
    if (n == 0 || n == 1) return 1
    return n * factorial(n - 1)
}

```



```

Введите число для вычисления факториала:
12
Факториал числа 12 = 479001600
Process finished with exit code 0

```

18.

```

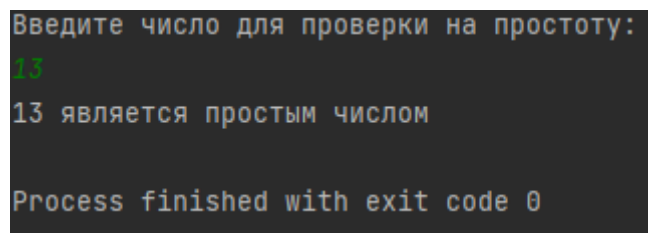
fun main() {
    println("Введите число для проверки на простоту:")
    val number = readLine()?.toIntOrNull()

    if (number != null && number > 0) {
        if (isPrime(number)) {
            println("$number является простым числом")
        } else {
            println("$number не является простым числом")
        }
    } else {
        println("Пожалуйста, введите корректное положительное число")
    }
}

```

```
}  
}
```

```
fun isPrime(number: Int): Boolean {  
    if (number <= 1) return false  
    if (number == 2) return true  
    if (number % 2 == 0) return false  
  
    val sqrt = kotlin.math.sqrt(number.toDouble()).toInt()  
    for (i in 3..sqrt step 2) {  
        if (number % i == 0) return false  
    }  
    return true  
}
```



```
Введите число для проверки на простоту:  
13  
13 является простым числом  
Process finished with exit code 0
```

19.

```
fun main() {  
    val numbers = arrayOf(1, 2, 3, 4, 5)  
    println("Массив: ${numbers.joinToString()}")  
    println("Сумма всех чисел: ${sumArray(numbers)}")  
  
    // Альтернативный ввод с клавиатуры
```

```

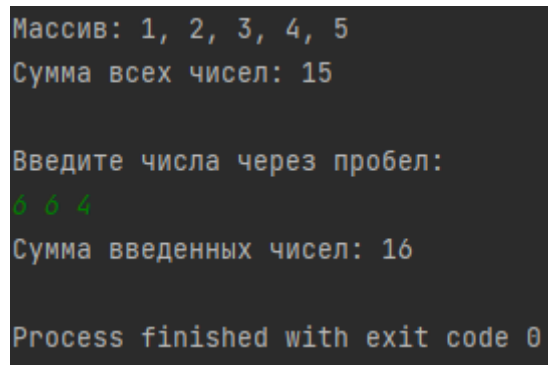
println("\nВведите числа через пробел:")
val input = readLine()
val userArray = input?.split(" ")?.mapNotNull { it.toIntOrNull() }?.toTypedArray()

if (userArray != null) {
    println("Сумма введенных чисел: ${sumArray(userArray)}")
}
}

fun sumArray(arr: Array<Int>): Int {
    return arr.sum()
}

// Альтернативная реализация с циклом
fun sumArrayWithLoop(arr: Array<Int>): Int {
    var sum = 0
    for (number in arr) {
        sum += number
    }
    return sum
}

```



```

Массив: 1, 2, 3, 4, 5
Сумма всех чисел: 15

Введите числа через пробел:
0 0 4
Сумма введенных чисел: 16

Process finished with exit code 0

```

```

fun main() {
    val numbers = arrayOf(12, 45, 3, 78, 23, 56, 9)
    println("Массив: ${numbers.joinToString()}")
    println("Максимальное число: ${findMax(numbers)}")

    // Вариант с вводом с клавиатуры
    println("\nВведите числа через пробел:")
    val input = readLine()
    val userArray = input?.split(" ")?.mapNotNull { it.toIntOrNull() }?.toTypedArray()

    if (userArray != null && userArray.isNotEmpty()) {
        println("Максимальное число среди введенных: ${findMax(userArray)}")
    } else {
        println("Введите корректные числа")
    }
}

fun findMax(arr: Array<Int>): Int {
    if (arr.isEmpty()) throw IllegalArgumentException("Массив пустой")
    return arr.maxOrNull() ?: throw IllegalArgumentException("Массив пустой")
}

// Альтернативная реализация с циклом
fun findMaxWithLoop(arr: Array<Int>): Int {
    if (arr.isEmpty()) throw IllegalArgumentException("Массив пустой")

    var max = arr[0]
    for (i in 1 until arr.size) {
        if (arr[i] > max) {
            max = arr[i]
        }
    }
}

```

```
    return max  
}
```

```
Массив: 12, 45, 3, 78, 23, 56, 9  
Максимальное число: 78  
  
Введите числа через пробел:  
55 555 11 9999  
Максимальное число среди введенных: 9999  
  
Process finished with exit code 0
```

21.

```
fun main() {  
    val numbers = arrayOf(64, 34, 25, 12, 22, 11, 90)  
    println("Исходный массив: ${numbers.joinToString()}")  
  
    val sortedArray = bubbleSort(numbers.clone())  
    println("Отсортированный массив: ${sortedArray.joinToString()}")  
  
    // Вариант с вводом с клавиатуры  
    println("\nВведите числа через пробел:")  
    val input = readLine()  
    val userArray = input?.split(" ")?.mapNotNull { it.toIntOrNull() }?.toTypedArray()  
  
    if (userArray != null && userArray.isNotEmpty()) {  
        val sortedUserArray = bubbleSort(userArray)  
        println("Отсортированный массив: ${sortedUserArray.joinToString()}")  
    }  
}
```

```
}
```

```
// Реализация сортировки пузырьком
```

```
fun bubbleSort(arr: Array<Int>): Array<Int> {  
    val n = arr.size  
    for (i in 0 until n - 1) {  
        for (j in 0 until n - i - 1) {  
            if (arr[j] > arr[j + 1]) {  
                // Обмен элементов  
                val temp = arr[j]  
                arr[j] = arr[j + 1]  
                arr[j + 1] = temp  
            }  
        }  
    }  
    return arr  
}
```

```
// Альтернативная реализация с использованием встроенной функции
```

```
fun sortArray(arr: Array<Int>): Array<Int> {  
    return arr.sortedArray()  
}
```

```
Исходный массив: 64, 34, 25, 12, 22, 11, 90  
Отсортированный массив: 11, 12, 22, 25, 34, 64, 90  
  
Введите числа через пробел:  
99 34 21 4 1 15  
Отсортированный массив: 1, 4, 15, 21, 34, 99  
  
Process finished with exit code 0
```

```

fun main() {
    println(isPalindrome("А роза упала на лапу Азора"))
    println(isPalindrome("Палиндром"))
}

fun isPalindrome(str: String): Boolean {
    val cleanStr = str.lowercase().replace(Regex("[^a-ya-z0-9]"), "")
    return cleanStr == cleanStr.reversed()
}

```

```

true
false

Process finished with exit code 0

```

23.

```

fun main() {
    val text = "Hello, World!"
    println(countCharacters(text)) // Выведет: 13
}

```

```

fun countCharacters(str: String): Int {
    return str.length
}

```

// Альтернативный вариант с исключением пробелов

```

fun countCharactersNoSpaces(str: String): Int {
    return str.replace(" ", "").length
}

```

```

13

Process finished with exit code 0

```

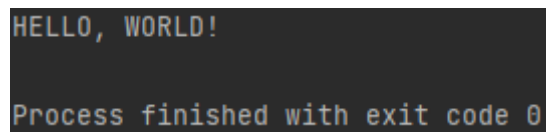

24.

```
fun main() {  
    val text = "Hello, World!"  
    println(convertToUpperCase(text)) // Выведет: HELLO, WORLD!  
}
```

```
fun convertToUpperCase(str: String): String {  
    return str.uppercase()  
}
```

// Альтернативный вариант с использованием toUpperCase()

```
fun convertToUpperCaseAlt(str: String): String {  
    return str.toUpperCase()  
}
```



```
HELLO, WORLD!  
  
Process finished with exit code 0
```

25.

```
fun main() {  
    val str1 = "Hello"  
    val str2 = "World"  
    println(concatenateStrings(str1, str2))  
}
```

```
fun concatenateStrings(str1: String, str2: String): String {  
    return str1 + str2  
}
```

```
HelloWorld
Process finished with exit code 0
```

26.

```
fun main() {
    val array = arrayOf(1, 2, 3, 4, 5, 6, 7, 8, 9)
    println(getLastElement(array)) // Выведет: 5

    // Проверка пустого массива
    val emptyArray = arrayOf<Int>()
    println(getLastElement(emptyArray)) // Выведет: null
}
```

```
fun <T> getLastElement(array: Array<T>): T? {
    return if (array.isNotEmpty()) {
        array.last()
    } else {
        null
    }
}
```

```
// Альтернативный вариант с использованием lastOrNull()
fun <T> getLastElementAlt(array: Array<T>): T? {
    return array.lastOrNull()
}
```

```
9
null
Process finished with exit code 0
```

27.

```
fun main() {
    val array = arrayOf(1, 2, 3, 4, 5)
```

```

println(containsElement(array, 3)) // Выведет: true
println(containsElement(array, 6)) // Выведет: false

// Пример с массивом строк
val stringArray = arrayOf("apple", "banana", "orange")
println(containsElement(stringArray, "banana")) // Выведет: true
println(containsElement(stringArray, "grape")) // Выведет: false
}

fun <T> containsElement(array: Array<T>, element: T): Boolean {
    return element in array
    // Альтернативные варианты:
    // return array.contains(element)
    // return array.any { it == element }
}

```

```

true
false
true
false

Process finished with exit code 0

```

28.

```

fun main() {
    println(createArray(5).joinToString()) // Выведет: 1, 2, 3, 4, 5
    println(createArray(3).joinToString()) // Выведет: 1, 2, 3

    // Альтернативный вызов
}

```

```

    val array = createArrayAlt(4)
    println(array.joinToString()) // Выведет: 1, 2, 3, 4
}

fun createArray(n: Int): Array<Int> {
    return Array(n) { it + 1 }
}

// Альтернативный вариант с использованием IntArray
fun createArrayAlt(n: Int): IntArray {
    return IntArray(n) { it + 1 }
}

// Еще один вариант с использованием диапазона
fun createArrayRange(n: Int): Array<Int> {
    return (1..n).toList().toTypedArray()
}

```

```

1, 2, 3, 4, 5
1, 2, 3
1, 2, 3, 4
Process finished with exit code 0

```

29.

```

fun main() {
    val array = arrayOf(5, 2, 8, 1, 9, 3)
    val (min, max) = findMinMax(array)
    println("Минимум: $min, Максимум: $max") // Выведет: Минимум: 1, Максимум: 9
}

```

```
// Проверка на пустой массив
val emptyArray = arrayOf<Int>()
val (emptyMin, emptyMax) = findMinMax(emptyArray)
println("Пустой массив: $emptyMin, $emptyMax") // Выведет: null, null
}
```

```
fun findMinMax(array: Array<Int>): Pair<Int?, Int?> {
    return if (array.isEmpty()) {
        Pair(null, null)
    } else {
        Pair(array.minOrNull(), array.maxOrNull())
    }
}
```

// Альтернативный вариант без использования стандартных функций

```
fun findMinMaxManual(array: Array<Int>): Pair<Int?, Int?> {
    if (array.isEmpty()) return Pair(null, null)

    var min = array[0]
    var max = array[0]

    for (element in array) {
        if (element < min) min = element
        if (element > max) max = element
    }

    return Pair(min, max)
}
```

```
Минимум: 1, Максимум: 9
Пустой массив: null, null

Process finished with exit code 0
```

30.

```
fun main() {
    println(calculateSum(5)) // Выведет: 15 (1 + 2 + 3 + 4 + 5)
    println(calculateSum(3)) // Выведет: 6 (1 + 2 + 3)
    println(calculateSum(100)) // Выведет: 5050
}
```

// Вариант 1: Использование арифметической прогрессии

```
fun calculateSum(n: Int): Long {
    return (n.toLong() * (n + 1)) / 2
}
```

```
15
6
5050

Process finished with exit code 0
```

31.

```
fun main() {
    println(celsiusToFahrenheit(0)) // Выведет: 32.0 (точка замерзания воды)
    println(celsiusToFahrenheit(100)) // Выведет: 212.0 (точка кипения воды)
    println(celsiusToFahrenheit(25)) // Выведет: 77.0 (комнатная температура)
    println(celsiusToFahrenheit(-40)) // Выведет: -40.0 (точка совпадения шкал)
}
```

```
fun celsiusToFahrenheit(celsius: Int): Double {
    return celsius * 9.0 / 5.0 + 32
}
```

// Вариант с округлением до определённого количества знаков после запятой

```
fun celsiusToFahrenheitRounded(celsius: Int, decimals: Int = 2): Double {
    return String.format("%.${decimals}f", celsius * 9.0 / 5.0 + 32).toDouble()
}
```

```
32.0
212.0
77.0
-40.0

Process finished with exit code 0
```

32.

```
fun main() {
    println(reverseString("Hello"))
    println(reverseString("World!"))
}
fun reverseString(str: String): String {
    return str.reversed()
}
```

```
oHlle
!dlroW

Process finished with exit code 0
```

33.

```

fun main() {
    val intArray = arrayOf(10, 20, 30, 40, 50)
    println(getElementByIndex(intArray, 2))
    println(getElementByIndex(intArray, 0))
    println(getElementByIndex(intArray, 4))
    println(getElementByIndex(intArray, 10))
    val stringArray = arrayOf("яблоко", "банан", "апельсин")
    println(getElementByIndex(stringArray, 1))
    println(getElementByIndex(stringArray, 5))
}
fun <T> getElementByIndex(array: Array<T>, index: Int): T? {
    return if (index in array.indices) array[index] else null
}

```

```

30
10
50
null
банан
null

Process finished with exit code 0

```

34.

```

fun main() {
    println(removeSpaces("Hello World"))
    println(removeSpaces(" Много пробелов "))
    println(removeSpaces("NoSpaces"))
    println(removeSpaces(" "))
    println(removeSpaces("Тест на русском языке"))
}
fun removeSpaces(str: String): String {
    return str.replace(" ", "")
}

```



```
HelloWorld
Многопробелов
NoSpaces

Тестнарусскомязыке

Process finished with exit code 0
```

35.

```
fun main() {
    println(sumNaturalNumbers(5))
    println(sumNaturalNumbers(10))
    println(sumNaturalNumbers(1))
    println(sumNaturalNumbers(100))
    println(sumNaturalNumbers(0))
}
fun sumNaturalNumbers(n: Int): Long {
    return (n.toLong() * (n + 1)) / 2
}
```

36.

```
fun main() {
    println(containsSubstring("Hello World", "World"))
    println(containsSubstring("Hello World", "Python"))
    println(containsSubstring("Привет мир", "мир"))
    println(containsSubstring("", "test"))
    println(containsSubstring("АБВГД", "БВ"))
}
fun containsSubstring(mainStr: String, subStr: String): Boolean {
    return mainStr.contains(subStr)
}
```

```
true
false
true
false
true

Process finished with exit code 0
```

37.

```
fun main() {
    print("Введите число: ")
    val number = readLine()?.toIntOrNull() ?: 0

    println("Таблица умножения для числа $number:")
    for (i in 1..10) {
        println("$number x $i = ${number * i}")
    }
}
```

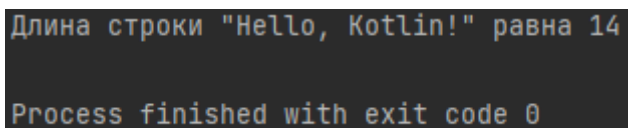
```
Введите число: 999
Таблица умножения для числа 999:
999 x 1 = 999
999 x 2 = 1998
999 x 3 = 2997
999 x 4 = 3996
999 x 5 = 4995
999 x 6 = 5994
999 x 7 = 6993
999 x 8 = 7992
999 x 9 = 8991
999 x 10 = 9990

Process finished with exit code 0
```

38.

```
fun stringLength(input: String): Int {  
    return input.length  
}
```

```
fun main() {  
    val testString = "Hello, Kotlin!"  
    val length = stringLength(testString)  
    println("Длина строки \"$testString\" равна $length")  
}
```



```
Длина строки "Hello, Kotlin!" равна 14  
Process finished with exit code 0
```

39.

40.

```
fun copyArray(original: IntArray): IntArray {  
    return original.copyOf()  
}
```

```
fun main() {  
    val originalArray = intArrayOf(1, 2, 3, 4, 5)  
    val copiedArray = copyArray(originalArray)  
  
    println("Original array: ${originalArray.joinToString()}")  
    println("Copied array: ${copiedArray.joinToString()}")  
}
```

```
// Изменяем оригинальный массив, чтобы показать, что копия не изменяется
originalArray[0] = 100

println("\nAfter modification:")
println("Original array: ${originalArray.joinToString()}")
println("Copied array: ${copiedArray.joinToString()}")
}
```

```
Original array: 1, 2, 3, 4, 5
Copied array: 1, 2, 3, 4, 5

After modification:
Original array: 100, 2, 3, 4, 5
Copied array: 1, 2, 3, 4, 5

Process finished with exit code 0
```

41.

```
fun countVowels(input: String): Int {
    val vowels = setOf('a', 'e', 'i', 'o', 'u', 'ä', 'ö', 'y', 'ı', 'и', 'э', 'я', 'ю', 'ё', 'е')
    var count = 0

    for (char in input.lowercase()) {
        if (char in vowels) {
            count++
        }
    }

    return count
}

fun main() {
    val testString = "Привет, мир! Hello, world!"
    val vowelCount = countVowels(testString)
```

```
println("Строка: \"\$testString\")
println("Количество гласных: \$vowelCount")
}
```

```
Строка: "Привет, мир! Hello, world!"
Количество гласных: 6

Process finished with exit code 0
```

42.

```
fun firstIndexOf(arr: IntArray, element: Int): Int {
    for (i in arr.indices) {
        if (arr[i] == element) {
            return i
        }
    }
    return -1
}

fun main() {
    val numbers = intArrayOf(4, 2, 8, 6, 2, 10)
    val element = 2

    val index = firstIndexOf(numbers, element)

    println("Массив: ${numbers.joinToString()}")
    println("Индекс первого вхождения элемента $element: $index")
    println("Индекс несуществующего элемента (99): ${firstIndexOf(numbers, 99)}")
}
```

```
Массив: 4, 2, 8, 6, 2, 10
Индекс первого вхождения элемента 2: 1
Индекс несуществующего элемента (99): -1

Process finished with exit code 0
```

