```kotlin
//1. Создание и вывод элементов

fun main() {
    val array = intArrayOf(1, 2, 3, 4, 5)
    println("Элементы массива: ${array.joinToString(", ")}")



//2. Сумма элементов массива

    val array1 = intArrayOf(1, 2, 3, 4, 5)
    val sum1 = array1.sum()
    println("Сумма элементов массива: $sum1")



//3. Максимальное и минимальное значение

    val array2 = intArrayOf(5, 3, 8, 1, 4, 10, 6, 2, 9, 7)
    val max = array2.maxOrNull()
    val min = array2.minOrNull()
    println("Максимальное значение: $max")
    println("Минимальное значение: $min")



//4. Сортировка массива

    val array3 = intArrayOf(5, 3, 8, 1, 4)
    val sortedArray = array3.sortedArray()
    println("Отсортированный массив: ${sortedArray.joinToString(", ")}")



//5. Уникальные элементы

    val array4 = intArrayOf(1, 2, 2, 3, 4, 4, 5)
    val uniqueElements = array4.distinct()
    println("Уникальные элементы: ${uniqueElements.joinToString(", ")}")



//6. Четные и нечетные числа

    val array5 = intArrayOf(1, 2, 3, 4, 5, 6)
    val evenNumbers = array5.filter { it % 2 == 0 }
    val oddNumbers = array5.filter { it % 2 != 0 }
    println("Четные числа: ${evenNumbers.joinToString(", ")}")
    println("Нечетные числа: ${oddNumbers.joinToString(", ")}")



//7. Реверс массива

    val array6 = intArrayOf(1, 2, 3, 4, 5)
    val reversedArray = array6.reversedArray()
    println("Реверсированный массив: ${reversedArray.joinToString(", ")}")



//8. Поиск элемента
```

```
    val array7 = intArrayOf(1, 2, 3, 4, 5)
    val elementToFind = 3
    val index = array7.indexOf(elementToFind)
    if (index != -1) {
        println("Элемент $elementToFind найден на индексе $index")
    } else {
        println("Элемент $elementToFind не найден")
    }



//9. Копирование массива


    val originalArray = intArrayOf(1, 2, 3)
    val copiedArray = originalArray.copyOf()
    println("Скопированный массив: ${copiedArray.joinToString(", ")}")


//10. Сумма четных чисел


    val array8 = intArrayOf(1, 2, 3, 4, 5)
    val sumOfEvens = array8.filter { it % 2 == 0 }.sum()
    println("Сумма четных чисел: $sumOfEvens")



//11. Пересечение массивов


    val array9 = intArrayOf(1, 2, 3, 4)
    val array10 = intArrayOf(3, 4, 5, 6)
    val intersection = array9.intersect(array10.toSet())
    println("Пересечение массивов: ${intersection.joinToString(", ")}")



//12. Перестановка элементов

fun swap(array11: IntArray, index1: Int, index2: Int) {
    val temp = array11[index1]
    array11[index1] = array11[index2]
    array11[index2] = temp
}


    val array12 = intArrayOf(1, 2, 3)
    swap(array, 0, 2)
    println("Массив после перестановки: ${array12.joinToString(", ")}")



//13. Заполнение случайными числами



    val randomArray = IntArray(20) { (Math.random()*101).toInt() }
    println("Случайные числа: ${randomArray.joinToString(", ")}")



//14. Числа Прокопенко (делятся на 3)
```

```
    val array14 = intArrayOf(1, 2, 3, 4, 5, 6)
    val divisibleByThree = array14.filter { it % 3 == 0 }
    println("Числа Прокопенко (делятся на 3):
${divisibleByThree.joinToString(", ")}")



//15. Проверка на палиндром

fun isPalindrome(array15: IntArray): Boolean {
    return array15.contentEquals(array15.reversedArray())
}

    val array15 = intArrayOf(1, 2, 3, 2, 1)
    println("Массив является палиндромом: ${isPalindrome(array15)}")



//16. Конкатенация двух массивов


    val array16 = intArrayOf(1, 2)
    val array17 = intArrayOf(3, 4)
    val concatenatedArray = array16 + array17
    println("Конкатенированный массив: ${concatenatedArray.joinToString(",
")}")



//17. Сумма и произведение




    val array18 = intArrayOf(1, 2, 3)
    val sum = array18.sum()
    val product = array18.fold(1) { acc, i -> acc * i }
    println("Сумма: $sum")
    println("Произведение: $product")



//18. Группировка чисел


    val array19 = intArrayOf(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)

    for (i in array19.indices step 5) {
        val group = if (i + 5 <= array19.size) {
            array19.slice(i until i + 5)
        } else {
            array19.slice(i until array19.size)
        }
        println("Группа: ${group.joinToString(", ")}")
    }



//19. Слияние двух массивов

fun mergeSortedArrays(array20: IntArray, array21: IntArray): IntArray {
    val mergedArray = IntArray(array1.size + array2.size)

    var i = 0 // индекс для первого массива
```

```
    var j = 0 // индекс для второго массива
    var k = 0 // индекс для объединенного массива

    while (i < array20.size && j < array21.size) {
        if (array20[i] < array21[j]) {
            mergedArray[k++] = array20[i++]
        } else {
            mergedArray[k++] = array21[j++]
        }
    }

    // Добавляем оставшиеся элементы из первого массива (если есть)
    while (i < array20.size) {
        mergedArray[k++] = array20[i++]
    }

    // Добавляем оставшиеся элементы из второго массива (если есть)
    while (j < array21.size) {
        mergedArray[k++] = array21[j++]
    }

    return mergedArray

    val sortedArray20 = intArrayOf(1, 3, 5)
    val sortedArray21 = intArrayOf(2, 4, 6)

    val mergedSortedArray = mergeSortedArrays(sortedArray20, sortedArray21)

    println("Слитый отсортированный массив:
${mergedSortedArray.joinToString(", ")}")

}
//| 20. Числовая последовательность (арифметическая прогрессия)

    val start = 1
    val difference = 2
    val count = 10
    val arithmeticProgression = IntArray(count) { start + it * difference }
    println("Арифметическая прогрессия:
${arithmeticProgression.joinToString(", ")}")


    //21. Удаление элемента

    fun removeElement(array22: IntArray, element: Int): IntArray {
        return array22.filter { it != element }.toIntArray()
    }

// Пример использования
    val arrayWithElement = intArrayOf(1, 2, 3, 4, 5)
    val updatedArray = removeElement(arrayWithElement, 3)
    println("Массив после удаления элемента: ${updatedArray.joinToString(",
")}")


    //22. Поиск второго максимального

    fun findSecondMaximum(array23: IntArray): Int? {
        val uniqueElements = array23.distinct().sortedDescending()
        return if (uniqueElements.size < 2) null else uniqueElements[1]
    }

// Пример использования
    val numbers = intArrayOf(5, 3, 8, 1, 4)
```

```
    val secondMax = findSecondMaximum(numbers)
    println("Второй по величине элемент: $secondMax")


    //23. Объединение массивов

    fun mergeArrays(vararg arrays: IntArray): IntArray {
        // Вычисляем общий размер для нового массива
        val totalLength = arrays.sumOf { it.size }
        val mergedArray = IntArray(totalLength)

        var currentIndex = 0

        // Копируем каждый массив в объединенный массив
        for (array24 in arrays) {
            for (element in array24) {
                mergedArray[currentIndex] = element
                currentIndex++
            }
        }

        return mergedArray
    }

// Пример использования
    val array25 = intArrayOf(1, 2, 3)
    val array26 = intArrayOf(4, 5)
    val array27 = intArrayOf(6, 7, 8)

    val mergedArray = mergeArrays(array25, array26, array27)
    println("Объединенный массив: ${mergedArray.joinToString(", ")}")



    //24. Транспонирование матрицы

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

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

    println("Транспонированная матрица:")
    transposedMatrix.forEach { println(it.joinToString(", ")) }


    //25. Линейный поиск

    fun linearSearch(array28: IntArray, target: Int): Boolean {
        for (element in array28) {
            if (element == target) return true
        }
        return false
    }

// Пример использования
    val searchArray = intArrayOf(1, 2, 3, 4, 5)
```

```
    val found = linearSearch(searchArray, 3)
    println("Элемент найден: $found")


    //26. Среднее арифметическое

    val numbersForAverage = intArrayOf(1, 2, 3, 4, 5)
    val average = numbersForAverage.average()
    println("Среднее арифметическое: $average")


    //27. Максимальная последовательность

    fun maxSequence(array29: IntArray): Pair<Int, Int> {
        var maxCount = 1
        var currentCount = 1

        for (i in 1 until array29.size) {
            if (array29[i] == array29[i - 1]) {
                currentCount++
            } else {
                maxCount = maxOf(maxCount, currentCount)
                currentCount = 1
            }
        }
        maxCount = maxOf(maxCount, currentCount)

        return maxCount to array29.size // Возвращаем максимальную длину
последовательности и размер массива для примера
    }

// Пример использования
    val sequenceArray = intArrayOf(1, 1, 2, 2, 2, 3, 4)
    val maxSeq = maxSequence(sequenceArray)
    println("Максимальная последовательность одинаковых элементов:
${maxSeq.first}")


    //28. Ввод и вывод массива

// Для ввода массива можно использовать Scanner или другой метод в
зависимости от среды выполнения.
// Здесь приведен пример с использованием статического массива.
    val userInputArray = intArrayOf(1, 2, 3, 4, 5)
    println("Введенный массив: ${userInputArray.joinToString(", ")}")


    //29. Нахождение медианы

    fun findMedian(array30: IntArray): Double {
        val sortedArray = array30.sorted()
        val size = sortedArray.size

        return if (size % 2 == 0) {
            (sortedArray[size / 2 - 1] + sortedArray[size / 2]) / 2.0
        } else {
            sortedArray[size / 2].toDouble()
        }
    }

// Пример использования
    val medianArray = intArrayOf(3, 1, 4, 2, 5)
    val medianValue = findMedian(medianArray)
    println("Медиана: $medianValue")
```

```
    //30. Распределение по группам

    val groupSize = 10
    val totalNumbers = 100
    val randomNumbers = IntArray(totalNumbers) { (1..100).random() }


    for (i in randomNumbers.indices step groupSize) {
        val group = randomNumbers.slice(i until minOf(i + groupSize,
randomNumbers.size))
        println("Группа ${i / groupSize + 1}: ${group.joinToString(", ")}")
    }

}
```