**Матрица**

1

fun main() {  
 // Создаем массив из 5 целых чисел  
 val numbers = *arrayOf*(1, 2, 3, 4, 5)  
  
 // Выводим элементы массива на экран  
 for (number in numbers) {  
 *println*(number)  
 }  
}

Ответ:

1

2

3

4

5

Process finished with exit code 0

2

fun main() {  
 // Инициализируем массив чисел  
 val numbers = *arrayOf*(1, 2, 3, 4, 5)  
  
 // Вычисляем сумму элементов массива  
 val sum = numbers.*sum*()  
  
 // Выводим сумму  
 *println*("Сумма элементов массива: $sum")  
}

Ответ:

Сумма элементов массива: 15

Process finished with exit code 0

3

fun main() {  
 // Создаем массив из 10 чисел  
 val numbers = *arrayOf*(5, 3, 8, 1, 2, 7, 4, 10, 6, 9)  
  
 // Находим максимальное и минимальное значение  
 val maxNumber = numbers.*maxOrNull*()  
 val minNumber = numbers.*minOrNull*()  
  
 // Выводим результаты  
 *println*("Максимальное значение: $maxNumber")  
 *println*("Минимальное значение: $minNumber")  
}

Ответ:

Максимальное значение: 10

Минимальное значение: 1

Process finished with exit code 0

4

fun bubbleSort(arr: IntArray): IntArray {  
 val n = arr.size  
 var swapped: Boolean  
  
 do {  
 swapped = false  
 for (i in 1 *until* n) {  
 if (arr[i - 1] > arr[i]) {  
 // Меняем места элементов  
 val temp = arr[i - 1]  
 arr[i - 1] = arr[i]  
 arr[i] = temp  
 swapped = true  
 }  
 }  
 } while (swapped)  
  
 return arr  
}  
  
fun main() {  
 val numbers = *intArrayOf*(64, 34, 25, 12, 22, 11, 90)  
 val sortedNumbers = *bubbleSort*(numbers)  
  
 *println*("Отсортированный массив: ${sortedNumbers.*joinToString*(", ")}")  
}

Ответ:

Отсортированный массив: 11, 12, 22, 25, 34, 64, 90

Process finished with exit code 0

5

fun main() {  
 val array = *arrayOf*(1, 2, 2, 3, 4, 4, 5, 5, 6) // Исходный массив  
  
 // Преобразуем массив в Set, чтобы получить уникальные значения  
 val uniqueElements = array.*toSet*()  
  
 // Выводим уникальные элементы в виде списка  
 *println*("Уникальные элементы: $uniqueElements")  
}

Ответ:

Уникальные элементы: [1, 2, 3, 4, 5, 6]

Process finished with exit code 0

6

fun main() {  
 // Исходный массив  
 val numbers = *arrayOf*(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)  
  
 // Массив для четных чисел  
 val evenNumbers = numbers.*filter* **{ it** % 2 == 0 **}**.*toTypedArray*()  
  
 // Массив для нечетных чисел  
 val oddNumbers = numbers.*filter* **{ it** % 2 != 0 **}**.*toTypedArray*()  
  
 // Вывод результатов  
 *println*("Четные числа: ${evenNumbers.*joinToString*(", ")}")  
 *println*("Нечетные числа: ${oddNumbers.*joinToString*(", ")}")  
}

Ответ:

Четные числа: 2, 4, 6, 8, 10

Нечетные числа: 1, 3, 5, 7, 9

Process finished with exit code 0

7

fun main() {  
 // Исходный массив  
 val array = *intArrayOf*(1, 2, 3, 4, 5)  
  
 // Выводим массив до реверса  
 *println*("Исходный массив: ${array.*joinToString*(", ")}")  
  
 // Реверсируем массив  
 val reversedArray = array.*reversedArray*()  
  
 // Выводим реверсированный массив  
 *println*("Реверсированный массив: ${reversedArray.*joinToString*(", ")}")  
}

Ответ:

Исходный массив: 1, 2, 3, 4, 5

Реверсированный массив: 5, 4, 3, 2, 1

Process finished with exit code 0

8

fun main() {  
 val array = *arrayOf*(5, 3, 7, 1, 9, 4) // пример массива  
 val elementToFind = 7 // элемент, который нужно найти  
  
 val index = *linearSearch*(array, elementToFind)  
  
 if (index != -1) {  
 *println*("Элемент $elementToFind найден на индексе $index.")  
 } else {  
 *println*("Элемент $elementToFind не найден в массиве.")  
 }  
}  
  
fun linearSearch(array: Array<Int>, element: Int): Int {  
 for (i in array.*indices*) {  
 if (array[i] == element) {  
 return i // возвращаем индекс элемента, если он найден  
 }  
 }  
 return -1 // возвращаем -1, если элемент не найден  
}

Ответ:

Элемент 7 найден на индексе 2.

Process finished with exit code 0

9

fun main() {  
 val originalArray = *arrayOf*(1, 2, 3, 4, 5)  
 val copiedArray = originalArray.*copyOf*()  
  
 *println*("Original Array: ${originalArray.*joinToString*()}")  
 *println*("Copied Array: ${copiedArray.*joinToString*()}")  
}

Ответ:

Original Array: 1, 2, 3, 4, 5

Copied Array: 1, 2, 3, 4, 5

Process finished with exit code 0

10

fun main() {  
 val array = *intArrayOf*(1, 2, 3, 4, 5, 6, 7, 8, 9, 10) // Исходный массив  
 val sumOfEvens = *sumOfEvenNumbers*(array)  
 *println*("Сумма четных чисел: $sumOfEvens")  
}  
  
fun sumOfEvenNumbers(arr: IntArray): Int {  
 var sum = 0  
 for (num in arr) {  
 if (num % 2 == 0) {  
 sum += num  
 }  
 }  
 return sum  
}

Ответ:

Сумма четных чисел: 30

Process finished with exit code 0

11

fun main() {  
 // Определение двух массивов  
 val array1 = *arrayOf*(1, 2, 3, 4, 5)  
 val array2 = *arrayOf*(4, 5, 6, 7, 8)  
  
 // Нахождение пересечения массивов  
 val intersection = array1.*intersect*(array2.*toSet*())  
  
 // Преобразование результата в массив (если необходимо)  
 val intersectionArray = intersection.*toTypedArray*()  
  
 // Вывод результата  
 *println*("Пересечение массивов: ${intersectionArray.*joinToString*(", ")}")  
}

Ответ:

Пересечение массивов: 4, 5

Process finished with exit code 0

12

fun <T> swap(array: Array<T>, index1: Int, index2: Int) {  
 if (index1 < 0 || index2 < 0 || index1 >= array.size || index2 >= array.size) {  
 throw IndexOutOfBoundsException("Неверные индексы: $index1 или $index2")  
 }  
 val temp = array[index1]  
 array[index1] = array[index2]  
 array[index2] = temp  
}  
  
fun main() {  
 val myArray = *arrayOf*(1, 2, 3, 4, 5)  
 *println*("Массив до перестановки: ${myArray.*joinToString*(", ")}")  
  
 *swap*(myArray, 1, 3) // Меняем местами элементы с индексами 1 и 3  
  
 *println*("Массив после перестановки: ${myArray.*joinToString*(", ")}")  
}

Ответ:

Массив до перестановки: 1, 2, 3, 4, 5

Массив после перестановки: 1, 4, 3, 2, 5

Process finished with exit code 0

13

import kotlin.random.Random  
  
fun main() {  
 // Создание массива из 20 случайных чисел от 1 до 100  
 val randomNumbers = IntArray(20) **{** Random.nextInt(1, 101) **}** // Вывод массива на экран  
 *println*("Случайные числа: ${randomNumbers.*joinToString*(", ")}")  
}

Ответ:

Случайные числа: 37, 50, 2, 54, 73, 17, 49, 28, 78, 80, 1, 16, 25, 30, 60, 19, 58, 73, 30, 6

Process finished with exit code 0

14

fun main() {  
 // Исходный массив чисел  
 val numbers = *arrayOf*(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12)  
  
 // Фильтруем и выводим числа, делящиеся на 3  
 val divisibleByThree = numbers.*filter* **{ it** % 3 == 0 **}** // Вывод результатов  
 *println*("Числа, делящиеся на 3: $divisibleByThree")  
}

Ответ:

Числа, делящиеся на 3: [3, 6, 9, 12]

Process finished with exit code 0

15

fun isPalindrome(array: IntArray): Boolean {  
 val n = array.size  
 for (i in 0 *until* n / 2) {  
 if (array[i] != array[n - i - 1]) {  
 return false  
 }  
 }  
 return true  
}  
  
fun main() {  
 val array = *intArrayOf*(1, 2, 3, 2, 1) // Пример массива  
 if (*isPalindrome*(array)) {  
 *println*("Массив является палиндромом.")  
 } else {  
 *println*("Массив не является палиндромом.")  
 }  
}

Ответ:

Массив является палиндромом.

Process finished with exit code 0

16

fun main() {  
 // Создаем два массива  
 val array1 = *arrayOf*(1, 2, 3)  
 val array2 = *arrayOf*(4, 5, 6)  
  
 // Конкатенация массивов  
 val concatenatedArray = array1 + array2  
  
 // Вывод результата  
 *println*(concatenatedArray.*joinToString*(", ")) // Выведет: 1, 2, 3, 4, 5, 6  
}

Ответ:

1, 2, 3, 4, 5, 6

Process finished with exit code 0

17

fun main() {  
 // Инициализируем массив  
 val array = *intArrayOf*(1, 2, 3, 4, 5)  
  
 // Вычисляем сумму и произведение  
 val sum = array.*sum*()  
 val product = array.*fold*(1) **{** acc, i **->** acc \* i **}** // Выводим результаты  
 *println*("Сумма: $sum")  
 *println*("Произведение: $product")  
}

Ответ:

Сумма: 15

Произведение: 120

Process finished with exit code 0

18

fun main() {  
 val numbers = (1..20).*toList*() // Пример массива чисел от 1 до 20  
 val groupedNumbers = numbers.*chunked*(5) // Группировка по 5 элементов  
  
 // Вывод групп  
 for (group in groupedNumbers) {  
 *println*(group)  
 }  
}

Ответ:

[1, 2, 3, 4, 5]

[6, 7, 8, 9, 10]

[11, 12, 13, 14, 15]

[16, 17, 18, 19, 20]

Process finished with exit code 0

19

fun mergeSortedArrays(array1: IntArray, array2: IntArray): IntArray {  
 val mergedArray = IntArray(array1.size + array2.size)  
 var index1 = 0  
 var index2 = 0  
 var mergedIndex = 0  
  
 // Сравнение элементов из обоих массивов и добавление наименьшего в mergedArray  
 while (index1 < array1.size && index2 < array2.size) {  
 if (array1[index1] <= array2[index2]) {  
 mergedArray[mergedIndex] = array1[index1]  
 index1++  
 } else {  
 mergedArray[mergedIndex] = array2[index2]  
 index2++  
 }  
 mergedIndex++  
 }  
  
 // Если остались элементы в первом массиве  
 while (index1 < array1.size) {  
 mergedArray[mergedIndex] = array1[index1]  
 index1++  
 mergedIndex++  
 }  
  
 // Если остались элементы во втором массиве  
 while (index2 < array2.size) {  
 mergedArray[mergedIndex] = array2[index2]  
 index2++  
 mergedIndex++  
 }  
  
 return mergedArray  
}  
  
fun main() {  
 val array1 = *intArrayOf*(1, 3, 5, 7)  
 val array2 = *intArrayOf*(2, 4, 6, 8)  
  
 val mergedArray = *mergeSortedArrays*(array1, array2)  
 *println*("Слитый массив: ${mergedArray.*joinToString*(", ")}")  
}

Ответ:

Слитый массив: 1, 2, 3, 4, 5, 6, 7, 8

Process finished with exit code 0

20

fun main() {  
 val start = 1 // Начальное значение  
 val step = 2 // Шаг прогрессии  
 val count = 10 // Количество элементов в массиве  
  
 val arithmeticProgression = IntArray(count) **{** start + **it** \* step **}** *println*("Арифметическая прогрессия: ${arithmeticProgression.*joinToString*()}")  
}

Ответ:

Арифметическая прогрессия: 1, 3, 5, 7, 9, 11, 13, 15, 17, 19

Process finished with exit code 0

21

fun removeElement(list: MutableList<Int>, element: Int): MutableList<Int> {  
 list.remove(element)  
 return list  
}  
  
fun main() {  
 val numbers = *mutableListOf*(1, 2, 3, 4, 5)  
 *println*("Исходный список: $numbers")  
  
 val elementToRemove = 3  
 val updatedList = *removeElement*(numbers, elementToRemove)  
  
 *println*("Обновленный список: $updatedList")  
}

Ответ:

Исходный список: [1, 2, 3, 4, 5]

Обновленный список: [1, 2, 4, 5]

Process finished with exit code 0

22

fun findSecondMax(arr: IntArray): Int? {  
 // Убираем дубликаты и сортируем массив  
 val distinctArr = arr.*distinct*().*sorted*()  
  
 // Проверяем, есть ли хотя бы два уникальных элемента  
 return if (distinctArr.size < 2) {  
 null // Если нет, возвращаем null  
 } else {  
 distinctArr[distinctArr.size - 2] // Возвращаем второй максимальный элемент  
 }  
}  
  
fun main() {  
 val array = *intArrayOf*(3, 5, 7, 2, 5, 2, 7, 8)  
 val secondMax = *findSecondMax*(array)  
  
 if (secondMax != null) {  
 *println*("Второй по величине элемент: $secondMax")  
 } else {  
 *println*("В массиве недостаточно уникальных элементов для нахождения второго максимального.")  
 }  
}

Ответ:

Второй по величине элемент: 7

Process finished with exit code 0

23

fun mergeArrays(vararg arrays: IntArray): List<Int> {  
 return arrays.*flatMap* **{ it**.*asIterable*() **}**}  
  
fun main() {  
 val array1 = *intArrayOf*(1, 2, 3)  
 val array2 = *intArrayOf*(4, 5, 6)  
 val array3 = *intArrayOf*(7, 8, 9)  
  
 val result = *mergeArrays*(array1, array2, array3)  
  
 *println*(result) // Вывод: [1, 2, 3, 4, 5, 6, 7, 8, 9]  
}

Ответ:

[1, 2, 3, 4, 5, 6, 7, 8, 9]

Process finished with exit code 0

24

fun main() {  
 // Исходная матрица  
 val matrix = *arrayOf*(  
 *intArrayOf*(1, 2, 3),  
 *intArrayOf*(4, 5, 6),  
 *intArrayOf*(7, 8, 9)  
 )  
  
 // Вывод исходной матрицы  
 *println*("Исходная матрица:")  
 *printMatrix*(matrix)  
  
 // Транспонирование матрицы  
 val transposedMatrix = *transpose*(matrix)  
  
 // Вывод транспонированной матрицы  
 *println*("Транспонированная матрица:")  
 *printMatrix*(transposedMatrix)  
}  
  
// Функция для транспонирования матрицы  
fun transpose(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  
}  
  
// Функция для вывода матрицы  
fun printMatrix(matrix: Array<IntArray>) {  
 for (row in matrix) {  
 for (value in row) {  
 *print*("$value ")  
 }  
 *println*()  
 }  
}

Ответ:

Исходная матрица:

1 2 3

4 5 6

7 8 9

Транспонированная матрица:

1 4 7

2 5 8

3 6 9

Process finished with exit code 0

25

fun linearSearch(array: IntArray, target: Int): Boolean {  
 for (element in array) {  
 if (element == target) {  
 return true  
 }  
 }  
 return false  
}  
  
fun main() {  
 val array = *intArrayOf*(1, 5, 2, 8, 3, 9, 4)  
 val target1 = 8  
 val target2 = 10  
  
 *println*("Элемент $target1 найден: ${*linearSearch*(array, target1)}") // true  
 *println*("Элемент $target2 найден: ${*linearSearch*(array, target2)}") // false  
}

Ответ:

Элемент 8 найден: true

Элемент 10 найден: false

Process finished with exit code 0

26

fun average(numbers: DoubleArray): Double {  
 if (numbers.*isEmpty*()) {  
 return 0.0 // или бросить исключение, если пустой массив недопустим  
 }  
 return numbers.*sum*() / numbers.size  
}  
  
fun main() {  
 val numbers = *doubleArrayOf*(1.0, 2.0, 3.0, 4.0, 5.0)  
 val avg = *average*(numbers)  
 *println*("Среднее арифметическое: $avg")  
  
  
 val emptyArray = *doubleArrayOf*()  
 val emptyAvg = *average*(emptyArray)  
 *println*("Среднее арифметическое пустого массива: $emptyAvg")  
  
 val mixedArray = *doubleArrayOf*(1.0, 2.5, 0.0, -1.5, 5.2)  
 val mixedAvg = *average*(mixedArray)  
 *println*("Среднее арифметическое смешанного массива: $mixedAvg")  
}

Ответ:

Среднее арифметическое: 3.0

Среднее арифметическое пустого массива: 0.0

Среднее арифметическое смешанного массива: 1.44

Process finished with exit code 0

27

fun maxSequence(array: IntArray): Int {  
 if (array.*isEmpty*()) return 0  
  
 var maxCount = 1  
 var currentCount = 1  
 for (i in 1 *until* array.size) {  
 if (array[i] == array[i - 1]) {  
 currentCount++  
 } else {  
 maxCount = *maxOf*(maxCount, currentCount)  
 currentCount = 1  
 }  
 }  
 return *maxOf*(maxCount, currentCount) //Проверяем последний отрезок  
}  
  
fun main() {  
 val array1 = *intArrayOf*(1, 1, 1, 2, 2, 3, 3, 3, 3, 4)  
 val array2 = *intArrayOf*(1, 2, 3, 4, 5)  
 val array3 = *intArrayOf*(1,1,1,1,1,1)  
 val array4 = *intArrayOf*() //Пустой массив  
  
 *println*("Максимальная последовательность в array1: ${*maxSequence*(array1)}") // 4  
 *println*("Максимальная последовательность в array2: ${*maxSequence*(array2)}") // 1  
 *println*("Максимальная последовательность в array3: ${*maxSequence*(array3)}") // 6  
 *println*("Максимальная последовательность в array4: ${*maxSequence*(array4)}") // 0  
  
}

Ответ:

Максимальная последовательность в array1: 4

Максимальная последовательность в array2: 1

Максимальная последовательность в array3: 6

Максимальная последовательность в array4: 0

Process finished with exit code 0

28

fun main() {  
 *print*("Введите количество элементов в массиве: ")  
 val n = *readLine*()?.*toIntOrNull*() ?: 0 //Обработка потенциально некорректного ввода  
  
 if (n <= 0) {  
 *println*("Количество элементов должно быть больше 0.")  
 return  
 }  
  
 val numbers = DoubleArray(n)  
 for (i in 0 *until* n) {  
 while (true) {  
 *print*("Введите элемент ${i + 1}: ")  
 val input = *readLine*()  
 val number = input?.*toDoubleOrNull*()  
 if (number != null) {  
 numbers[i] = number  
 break  
 } else {  
 *println*("Некорректный ввод. Попробуйте ещё раз.")  
 }  
 }  
 }  
  
 *println*("Вы ввели массив: ${numbers.*contentToString*()}")  
}

Ответ:

Введите количество элементов в массиве: 3

Введите элемент 1: 2

Введите элемент 2: 7

Введите элемент 3: 8

Вы ввели массив: [2.0, 7.0, 8.0]

Process finished with exit code 0

29

fun median(array: DoubleArray): Double {  
 if (array.*isEmpty*()) return 0.0 // Обработка пустого массива  
  
 val sortedArray = array.*sortedArray*()  
 val mid = sortedArray.size / 2  
 return if (sortedArray.size % 2 == 1) {  
 sortedArray[mid] // Нечетное количество элементов  
 } else {  
 (sortedArray[mid - 1] + sortedArray[mid]) / 2.0 // Четное количество элементов  
 }  
}  
  
fun main() {  
 val array1 = *doubleArrayOf*(1.0, 3.0, 5.0, 2.0, 4.0)  
 val array2 = *doubleArrayOf*(1.0, 3.0, 5.0, 2.0, 4.0, 6.0)  
 val array3 = *doubleArrayOf*() //Пустой массив  
  
 *println*("Медиана array1: ${*median*(array1)}") // 3.0  
 *println*("Медиана array2: ${*median*(array2)}") // 3.5  
 *println*("Медиана array3: ${*median*(array3)}") // 0.0  
}

Ответ:

Медиана array1: 3.0

Медиана array2: 3.5

Медиана array3: 0.0

Process finished with exit code 0

30

fun groupNumbers(numbers: IntArray): List<IntArray> {  
 if (numbers.size != 100) {  
 throw IllegalArgumentException("Массив должен содержать 100 элементов.")  
 }  
 val groups = *mutableListOf*<IntArray>()  
 for (i in 0 *until* 10) {  
 val group = IntArray(10)  
 for (j in 0 *until* 10) {  
 group[j] = numbers[i \* 10 + j]  
 }  
 groups.add(group)  
 }  
 return groups  
}  
  
fun main() {  
 val numbers = IntArray(100) **{ it** + 1 **}** //Создаем массив из 100 чисел от 1 до 100  
 val groups = *groupNumbers*(numbers)  
  
 for (i in groups.*indices*) {  
 *println*("Группа ${i + 1}: ${groups[i].*contentToString*()}")  
 }  
}

Ответ:

Группа 1: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

Группа 2: [11, 12, 13, 14, 15, 16, 17, 18, 19, 20]

Группа 3: [21, 22, 23, 24, 25, 26, 27, 28, 29, 30]

Группа 4: [31, 32, 33, 34, 35, 36, 37, 38, 39, 40]

Группа 5: [41, 42, 43, 44, 45, 46, 47, 48, 49, 50]

Группа 6: [51, 52, 53, 54, 55, 56, 57, 58, 59, 60]

Группа 7: [61, 62, 63, 64, 65, 66, 67, 68, 69, 70]

Группа 8: [71, 72, 73, 74, 75, 76, 77, 78, 79, 80]

Группа 9: [81, 82, 83, 84, 85, 86, 87, 88, 89, 90]

Группа 10: [91, 92, 93, 94, 95, 96, 97, 98, 99, 100]

Process finished with exit code 0