Лабораторная работа №6

Ермолаева Н.Н

```
fun main() {
    val numbers = arrayOf(7, 5, 3, 9, 1)
    for (number in numbers) {
        println(number)
    }
}
```

```
fun main() {
    val numbers = arrayOf(7, 5, 3, 9, 4)
    var sum = 0
    for (number in numbers) {
        sum += number
    }
    println("Сумма элементов массива: $sum")
}
```

```
fun main() {
    val numbers = arrayOf(5, 7, 8, 3, 2, 6, 1, 4, 9, 2)
    var max = numbers[0]
    var min = numbers[0]
    for (num in numbers) {
        if (num > max) {
            max = num
        }
        if (num < min) {
            min = num
        }
    }
    println("Максимальное значение: $max")
    println("Минимальное значение: $min")
}
```

```kotlin
fun main() {
    val numbers = arrayOf(2, 4, 6, 8, 1, 3, 5, 7, 23, 10)
    for (i in 0 ≤ until < numbers.size - 1) {
        for (j in 0 ≤ until < numbers.size - i - 1) {
            if (numbers[j] > numbers[j + 1]) {
                val temp = numbers[j]
                numbers[j] = numbers[j + 1]
                numbers[j + 1] = temp
            }
        }
    }
    println("Отсортированный массив: ${numbers.joinToString( separator: ", ")}")
}
```

```kotlin
fun main() {
    val numbers = arrayOf(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
    val evenNumbers = numbers.filter { it % 2 == 0 }
    val oddNumbers = numbers.filter { it % 2 != 0 }
    println("Чётные числа: ${evenNumbers.joinToString( separator: ", ")}")
    println("Нечётные числа: ${oddNumbers.joinToString( separator: ", ")}")
}
```

```kotlin
fun main() {
    val numbers = arrayOf(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
    val reversedNumbers = numbers.reversedArray()
    println("Исходный массив: ${numbers.joinToString( separator: ", ")}")
    println("Реверсированный массив: ${reversedNumbers.joinToString( separator: ", ")}")
}
```

```kotlin
fun main() {
    val numbers = arrayOf(65, 73, 86, 37, 30)
    val target = 30
    val index = numbers.indexOf(target)
    if (index != -1) {
        println("Элемент $target найден по индексу $index")
    } else {
        println("Элемент $target не найден в массиве")
    }
}
```

```kotlin
fun main() {
    val originalArray = arrayOf(3, 45, 8, 42, 545)
    val newArray = originalArray.copyOf()
    println("Исходный массив: ${originalArray.joinToString( separator: ", ")}")
    println("Новый массив: ${newArray.joinToString( separator: ", ")}")
}
```

```kotlin
fun main() {
    val numbers = arrayOf(1, 2, 3, 4, 5, 6, 7, 8, 9)
    val sumEven = numbers.filter { it % 2 == 0 }.sum()
    println("Сумма четных чисел: $sumEven")
}
```

```kotlin
fun main() {
    val array1 = arrayOf(1, 2, 3, 4, 5)
    val array2 = arrayOf(6, 7, 2, 1, 8)
    val intersection = array1.intersect(array2.asIterable()).toList()
    println("Пересечение массивов: $intersection")
}
```

ъ

```kotlin
fun swapElements(array: IntArray, index1: Int, index2: Int) {
    if (index1 < 0 || index1 >= array.size || index2 < 0 || index2 >= array.size) {
        println("Индексы находятся вне диапазона массива.")
        return
    }
    val temp = array[index1]
    array[index1] = array[index2]
    array[index2] = temp
}
fun main() {
    val array = intArrayOf(6, 1, 4, 3, 2)
    println("Исходный массив: ${array.joinToString( separator: ", ")}")

    swapElements(array, index1: 1 , index2: 2)

    println("Массив после перестановки: ${array.joinToString( separator: ", ")}")
}

```

```kotlin
import kotlin.random.Random
fun main() {
    val array = IntArray( size: 20) { Random.nextInt( from: 1, until: 101) }
    println("Массив случайных чисел: ${array.joinToString( separator: ", ")}")
}

```

```kotlin
fun main() {
    val numbers = arrayOf(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
    println("Числа, делящиеся на 3:")
    for (number in numbers) {
        if (number % 3 == 0) {
            println(number)
        }
    }
}
```

```kotlin
fun main() {
    val array = arrayOf(1, 2, 3, 2, 1)
    if (isPalindrome(array)) {
        println("Массив является палиндромом.")
    } else {
        println("Массив не является палиндромом.")
    }
}
fun isPalindrome(array: Array<Int>): Boolean {
    val reversedArray = array.reversedArray()
    return array.contentEquals(reversedArray)
}
```

```kotlin
fun main() {
    val array1 = arrayOf(1, 2, 3)
    val array2 = arrayOf(3, 2, 1)
    val combinedArray = array1 + array2
    println("Соединенный массив: ${combinedArray.joinToString( separator: ", ")}")
}
```

```kotlin
fun main() {
    val array = arrayOf(4, 6, 2, 8, 5)
    val sum = array.sum()
    val product = array.fold( initial: 1) { acc, i -> acc * i }
    println("Сумма элементов массива: $sum")
    println("Произведение элементов массива: $product")
}
```

```kotlin
fun main() {
    val array = (1 ≤ .. ≤ 10).toList()
    val groupedArray = array.chunked( size: 5)
    for (group in groupedArray) {
        println(group) // Вывод каждой группы
    }
}
```

```kotlin
fun main() {
    val array1 = intArrayOf(1, 3, 5, 7, 9)
    val array2 = intArrayOf(2, 4, 6, 8, 10)
    val mergedArray = mergeSortedArrays(array1, array2)
    println(mergedArray.joinToString( separator: ", ")) // Вывод результата
}

fun mergeSortedArrays(arr1: IntArray, arr2: IntArray): IntArray {
    return (arr1 + arr2).sortedArray() // Сливаем и сортируем
}

```

```kotlin
fun main() {
    val start = 1
    val difference = 2
    val size = 10
    val arithmeticProgression = IntArray(size) { start + it * difference }
    println(arithmeticProgression.joinToString( separator: ", "))
}
```

```kotlin
fun removeElement(array: IntArray, elementToRemove: Int): IntArray {
    return array.filter { it != elementToRemove }.toIntArray()
}
fun main() {
    val array = intArrayOf(1, 2, 3, 4, 5, 3, 6, 7, 8, 9, 10)
    val elementToRemove = 3
    val updatedArray = removeElement(array, elementToRemove)
    println(updatedArray.joinToString( separator: ", "))
}
```

```kotlin
fun findSecondMaximum(array: IntArray): Int? {
    val uniqueValues = array.toSet()
    val sortedValues = uniqueValues.sortedDescending()

    return if (sortedValues.size > 1) sortedValues[1] else null
}
fun main() {
    val array = intArrayOf(1, 3, 7, 5, 9, 10, 11, 12)
    val secondMax = findSecondMaximum(array)
    if (secondMax != null) {
        println("Второй по величине элемент: $secondMax")
    } else {
        println("Второго по величине элемента нет")
    }
}
```

```kotlin
fun main() {
    val array1 = arrayOf(1, 2, 3)
    val array2 = arrayOf(4, 5, 6)
    val array3 = arrayOf(7, 8, 9)
    val resultArray = mergeArrays(array1, array2, array3)
    println("Результирующий массив: ${resultArray.joinToString()}")
}
fun mergeArrays(vararg arrays: Array<Int>): Array<Int> {
    return arrays.flatten().toTypedArray()
}
```

```kotlin
fun main() {
    val matrix = arrayOf(
        intArrayOf(1, 2, 3),
        intArrayOf(4, 5, 6),
        intArrayOf(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]
        }
    }
    println("Исходная матрица:")
    matrix.forEach { println(it.joinToString( separator: " ")) }
    println("Транспонированная матрица:")
    transposed.forEach { println(it.joinToString( separator: " ")) }
}
```

```kotlin
fun linearSearch(array: IntArray, target: Int): Boolean {
    for (element in array) {
        if (element == target) {
            return true
        }
    }
    return false
}
fun main() {
    val array = intArrayOf(44, 33, 22, 55, 66)
    val target = 66
    if (linearSearch(array, target)) {
        println("Элемент $target найден в массиве.")
    } else {
        println("Элемент $target не найден в массиве.")
    }
}
```

```kotlin
fun calculateAverage(array: IntArray): Double {
    if (array.isEmpty()) {
        return 0.0
    }
    val sum = array.sum()
    return sum.toDouble() / array.size
}
fun main() {
    val array = intArrayOf(54, 27, 32, 87, 60)
    val average = calculateAverage(array)
    println("Среднее арифметическое всех чисел в массиве: $average")
}
```

```kotlin
fun findMaxSequence(array: IntArray): Pair<Int, Int> {
    if (array.isEmpty()) return Pair(0, 0)
    var maxCount = 1
    var currentCount = 1
    var maxElement = array[0]
    for (i in 1 until array.size) {
        if (array[i] == array[i - 1]) {
            currentCount++
        } else {
            if (currentCount > maxCount) {
                maxCount = currentCount
                maxElement = array[i - 1]
            }
            currentCount = 1
        }
    }
    if (currentCount > maxCount) {
        maxCount = currentCount
        maxElement = array.last()
    }

    return Pair(maxElement, maxCount)
}
fun main() {
    val array = intArrayOf(7, 7, 7, 5, 3, 4, 2, 1)
    val (element, count) = findMaxSequence(array)
    println("Элемент $element встречается $count раз подряд")
}
```

```kotlin
fun main() {
    println("Введите количество элементов массива:")
    val size = readLine()?.toIntOrNull() ?: return
    val array = IntArray(size)
    println("Введите $size чисел (разделяйте пробелами):")
    val input = readLine() ?: return
    val numbers = input.split( delimiters: " ").mapNotNull { it.toIntOrNull() }
    if (numbers.size != size) {
        println("Количество введенных чисел не совпадает с заданным размером массива.")
        return
    }
    for (i in numbers.indices) {
        array[i] = numbers[i]
    }
    println("Ваш массив:")
    println(array.joinToString( separator: ", "))
}
```

```kotlin
fun main() {
    println("Введите числа, разделенные пробелами:")
    val input = readLine() ?: return
    val numbers = input.split( delimiters: " ").mapNotNull { it.toIntOrNull() }.sorted()
    val size = numbers.size
    if (size == 0) {
        println("Массив пустой.")
        return
    }
    val median = if (size % 2 == 1) {
        numbers[size / 2]
    } else {
        (numbers[size / 2 - 1] + numbers[size / 2]) / 2.0
    }
    println("Медиана: $median")
}
```

```kotlin
fun main() {
    val numbers = IntArray( size: 100) { (1 + .. + 100).random() }
    val groups = numbers.toList().chunked( size: 10)
    for (i in groups.indices) {
        println("Группа ${i + 1}: ${groups[i]}")
    }
}
```