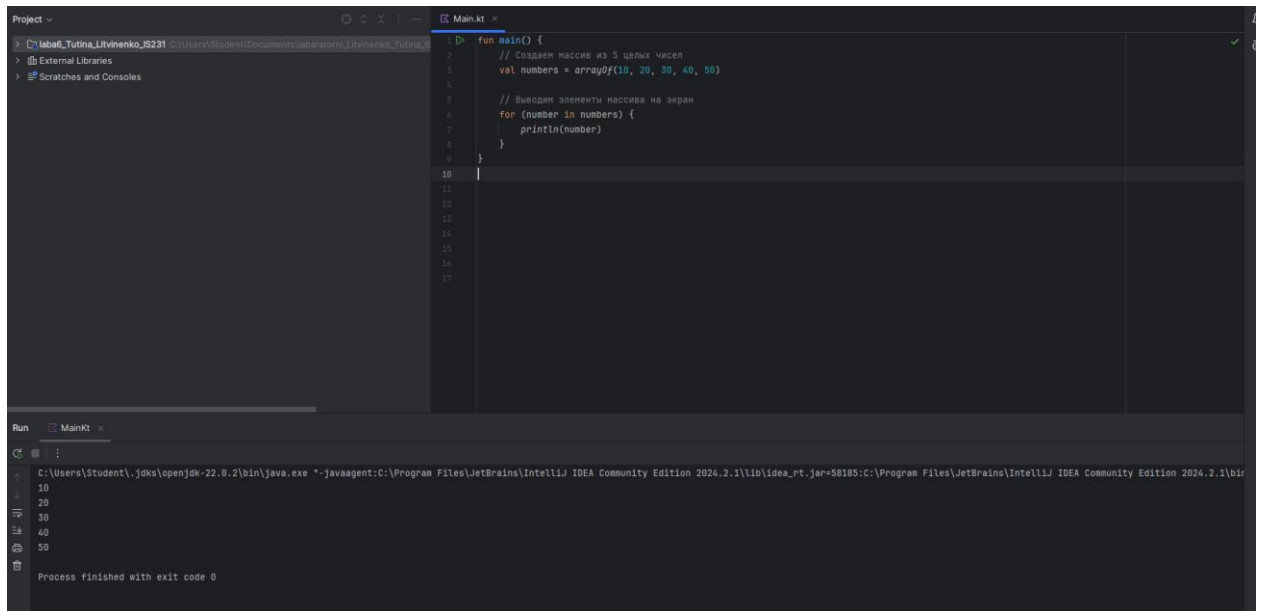


## Практическая работа №6

1.

```
fun main() {  
    // Создаем массив из 5 целых чисел  
    val numbers = arrayOf(10, 20, 30, 40, 50)  
  
    // Выводим элементы массива на экран  
    for (number in numbers) {  
        println(number)  
    }  
}
```



2.

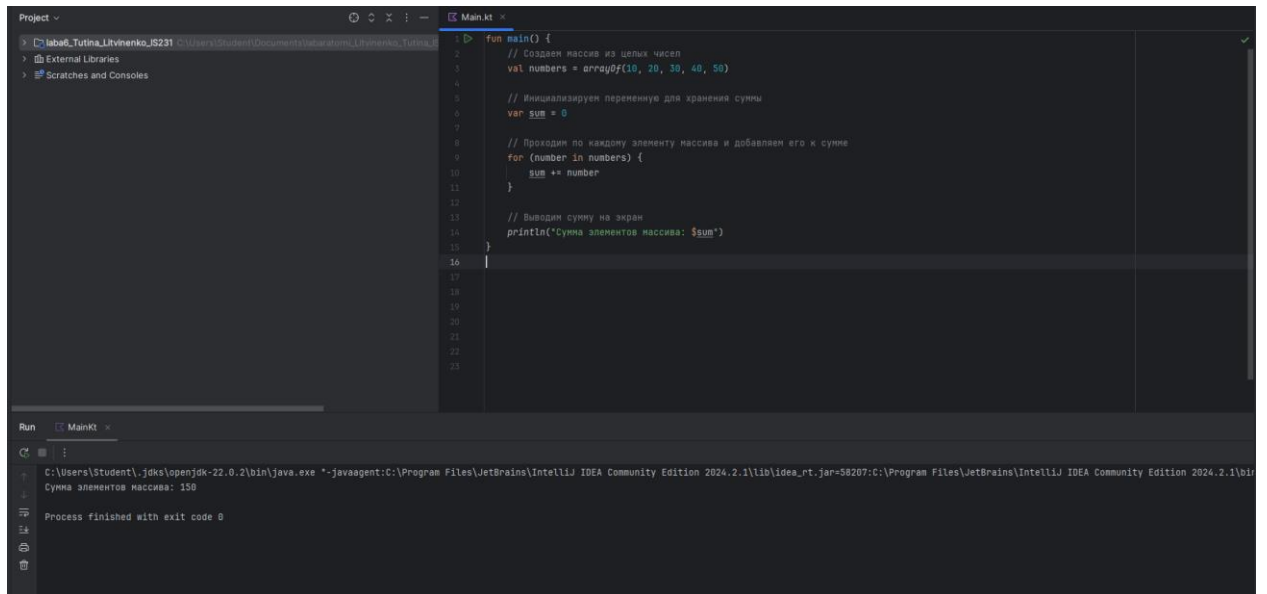
```
fun main() {  
    // Создаем массив из целых чисел  
    val numbers = arrayOf(10, 20, 30, 40, 50)  
  
    // Инициализируем переменную для хранения суммы  
    var sum = 0  
  
    // Проходим по каждому элементу массива и добавляем его к сумме  
    for (number in numbers) {
```

```

        sum += number
    }

    // Выводим сумму на экран
    println("Сумма элементов массива: $sum")
}

```



3.

```

fun main() {
    // Создаем массив из 10 целых чисел
    val numbers = arrayOf(15, 42, 8, 23, 4, 16, 32, 7, 19, 25)

    // Инициализируем переменные для хранения максимального и минимального значений
    var max = numbers[0]
    var min = numbers[0]

    // Проходим по каждому элементу массива
    for (number in numbers) {
        // Находим максимальное значение
        if (number > max) {
            max = number
        }

        // Находим минимальное значение

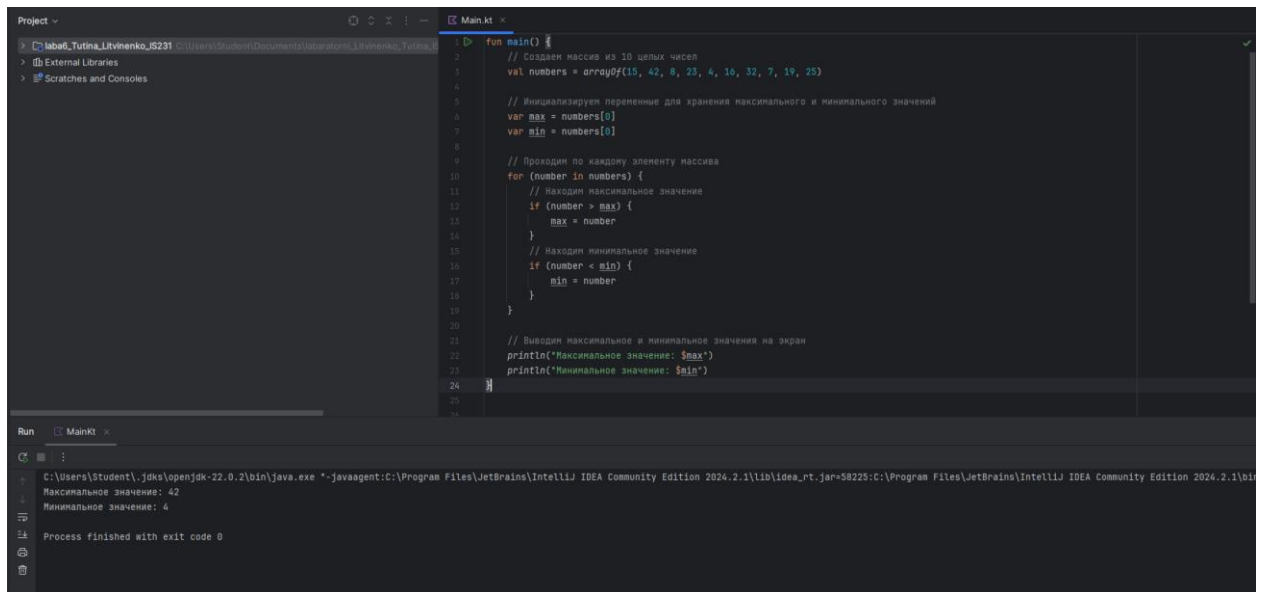
```

```

        if (number < min) {
            min = number
        }
    }
}

// Выводим максимальное и минимальное значения на экран
println("Максимальное значение: $max")
println("Минимальное значение: $min")
}.

```



```

1 fun main() {
2     // Создаем массив из 10 целых чисел
3     val numbers = arrayOf(15, 42, 8, 23, 4, 16, 32, 7, 19, 25)
4
5     // Инициализируем переменные для хранения максимального и минимального значений
6     var max = numbers[0]
7     var min = numbers[0]
8
9     // Проходим по каждому элементу массива
10    for (number in numbers) {
11        // Находим максимальное значение
12        if (number > max) {
13            max = number
14        }
15        // Находим минимальное значение
16        if (number < min) {
17            min = number
18        }
19    }
20
21    // Выводим максимальное и минимальное значения на экран
22    println("Максимальное значение: $max")
23    println("Минимальное значение: $min")
24 }

```

Run MainKt

```

C:\Users\Student\jdk\openjdk-22.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.2.1\lib\idea_rt.jar=58225:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.2.1\bin"
Максимальное значение: 42
Минимальное значение: 4
Process finished with exit code 0

```

4.

```

fun main() {
    // Создаем массив целых чисел
    val numbers = arrayOf(15, 42, 8, 23, 4, 16, 32, 7, 19, 25)

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

    // Сортируем массив с помощью алгоритма пузырька
    bubbleSort(numbers)

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

```

```
}
```

// Функция для сортировки массива пузырьком

```
fun bubbleSort(arr: 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]) {
```

```
                // Меняем местами элементы
```

```
                val temp = arr[j]
```

```
                arr[j] = arr[j + 1]
```

```
                arr[j + 1] = temp
```

```
            }
```

```
        }
```

```
    }
```

```
}
```

The screenshot shows an IDE with a Kotlin file named Main.kt. The code defines a `bubbleSort` function and a `main` function. The `main` function creates an array of integers: `val numbers = arrayOf(15, 42, 8, 23, 4, 16, 32, 7, 19, 25)`. It prints the initial array, calls `bubbleSort(numbers)`, and prints the sorted array. The `bubbleSort` function is implemented as shown in the previous blocks. The Run console at the bottom shows the output: "Исходный массив: 15, 42, 8, 23, 4, 16, 32, 7, 19, 25" and "Отсортированный массив: 4, 7, 8, 15, 16, 19, 23, 25, 32, 42". The process finished with exit code 0.

5.

```
fun main() {
```

```
    // Создаем массив целых чисел с дубликатами
```

```
    val numbers = arrayOf(15, 42, 8, 23, 15, 4, 16, 32, 7, 19, 4, 25)
```

```

// Выводим исходный массив

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

// Получаем уникальные элементы

val uniqueNumbers = getUniqueElements(numbers)

// Выводим уникальные элементы

println("Уникальные элементы: ${uniqueNumbers.joinToString(", ")}")
}

// Функция для получения уникальных элементов из массива
fun getUniqueElements(arr: Array<Int>): Set<Int> {
    return arr.toSet() // Преобразуем массив в множество
}

```

The screenshot shows an IDE window with a Kotlin file named 'Main.kt'. The code defines a function `getUniqueElements` and a `main` function. The `main` function creates an array of integers with duplicates, prints it, calls `getUniqueElements`, and prints the resulting set of unique elements. The output window at the bottom shows the execution results: 'Исходный массив: 15, 42, 8, 23, 15, 4, 16, 32, 7, 19, 4, 25' and 'Уникальные элементы: 15, 42, 8, 23, 4, 16, 32, 7, 19, 25'. The process finished with exit code 0.

```

Project: lab06_Tutina_Litvinenko_JS231
Main.kt
1 fun main() {
2     // Создаем массив целых чисел с дубликатами
3     val numbers = arrayOf(15, 42, 8, 23, 15, 4, 16, 32, 7, 19, 4, 25)
4
5     // Выводим исходный массив
6     println("Исходный массив: ${numbers.joinToString(", ")}")
7
8     // Получаем уникальные элементы
9     val uniqueNumbers = getUniqueElements(numbers)
10
11    // Выводим уникальные элементы
12    println("Уникальные элементы: ${uniqueNumbers.joinToString(", ")}")
13 }
14
15 // Функция для получения уникальных элементов из массива
16 fun getUniqueElements(arr: Array<Int>): Set<Int> {
17     return arr.toSet() // Преобразуем массив в множество
18 }
19
20
21
22
23
24
25
26
27
Run: Main.kt
C:\Users\Student\jdk\openjdk-22.0.2\bin\java.exe -javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.2.1\lib\idea_rt.jar=58258:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.2.1\bin
Исходный массив: 15, 42, 8, 23, 15, 4, 16, 32, 7, 19, 4, 25
Уникальные элементы: 15, 42, 8, 23, 4, 16, 32, 7, 19, 25
Process finished with exit code 0

```

6.

```

fun main() {
    // Создаем массив целых чисел

    val numbers = arrayOf(15, 42, 8, 23, 15, 4, 16, 32, 7, 19, 4, 25)

    // Выводим исходный массив

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

```

// Инициализируем списки для четных и нечетных чисел

```
val evenNumbers = mutableListOf<Int>()
```

```
val oddNumbers = mutableListOf<Int>()
```

// Разделяем числа на четные и нечетные

```
for (number in numbers) {  
    if (number % 2 == 0) {  
        evenNumbers.add(number) // Добавляем четное число  
    } else {  
        oddNumbers.add(number) // Добавляем нечетное число  
    }  
}
```

// Преобразуем списки в массивы

```
val evenArray = evenNumbers.toArray()
```

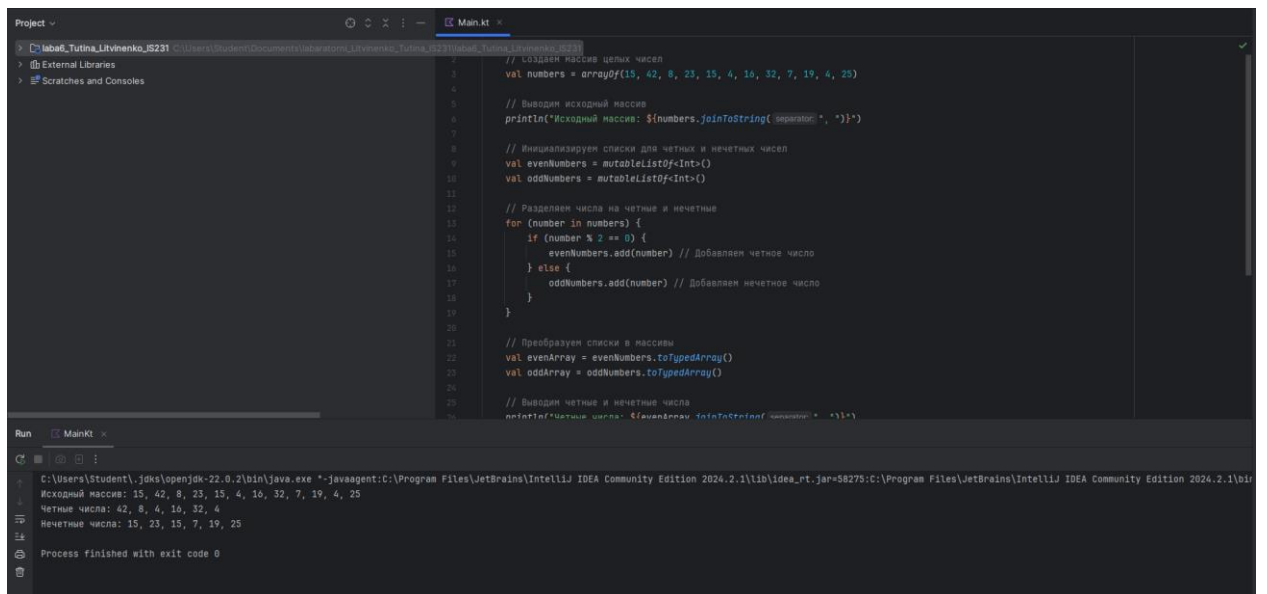
```
val oddArray = oddNumbers.toArray()
```

// Выводим четные и нечетные числа

```
println("Четные числа: ${evenArray.joinToString(", ")}")
```

```
println("Нечетные числа: ${oddArray.joinToString(", ")}")
```

```
}
```



The screenshot displays the IntelliJ IDEA IDE with a Kotlin file named 'Main.kt'. The code implements a program to separate even and odd numbers from an initial array. The initial array is [15, 42, 8, 23, 15, 4, 10, 32, 7, 19, 4, 25]. The program uses mutable lists to collect even and odd numbers, then converts them to arrays and prints them. The output window shows the execution results: 'Исходный массив: 15, 42, 8, 23, 15, 4, 10, 32, 7, 19, 4, 25', 'Четные числа: 42, 8, 4, 10, 32, 4', and 'Нечетные числа: 15, 23, 15, 7, 19, 25'. The process finished with exit code 0.

```
1 // Создаем массив целых чисел  
2 val numbers = arrayOf(15, 42, 8, 23, 15, 4, 10, 32, 7, 19, 4, 25)  
3  
4  
5 // Выводим исходный массив  
6 println("Исходный массив: ${numbers.joinToString(", ")}")  
7  
8  
9 // Инициализируем списки для четных и нечетных чисел  
10 val evenNumbers = mutableListOf<Int>()  
11 val oddNumbers = mutableListOf<Int>()  
12  
13 // Разделяем числа на четные и нечетные  
14 for (number in numbers) {  
15     if (number % 2 == 0) {  
16         evenNumbers.add(number) // Добавляем четное число  
17     } else {  
18         oddNumbers.add(number) // Добавляем нечетное число  
19     }  
20 }  
21  
22 // Преобразуем списки в массивы  
23 val evenArray = evenNumbers.toArray()  
24 val oddArray = oddNumbers.toArray()  
25  
26 // Выводим четные и нечетные числа  
27 println("Четные числа: ${evenArray.joinToString(", ")}")  
28 println("Нечетные числа: ${oddArray.joinToString(", ")}")  
29  
30 }
```

Run Main.kt  
C:\Users\Student\jdk\openjdk-22.0.2\bin\java.exe -javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.2.1\lib\idea\_rt.jar=58275:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.2.1\bin  
Исходный массив: 15, 42, 8, 23, 15, 4, 10, 32, 7, 19, 4, 25  
Четные числа: 42, 8, 4, 10, 32, 4  
Нечетные числа: 15, 23, 15, 7, 19, 25  
Process finished with exit code 0

```

fun main() {

    // Создаем массив целых чисел

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


    // Выводим исходный массив

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


    // Реверсируем массив

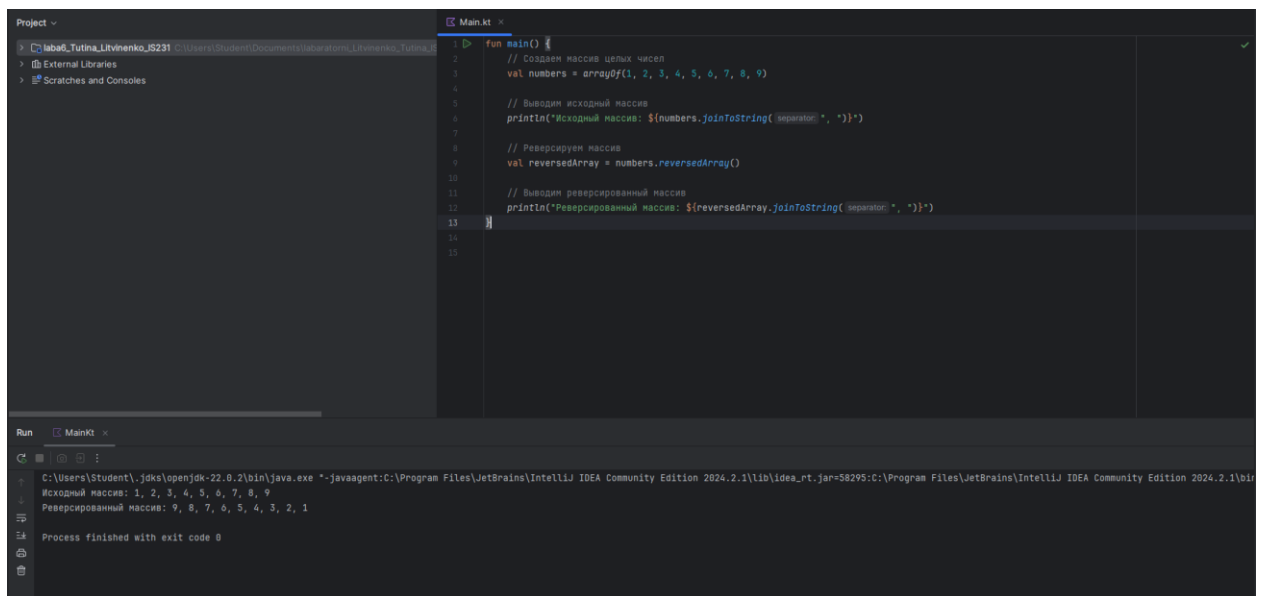
    val reversedArray = numbers.reversedArray()


    // Выводим реверсированный массив

    println("Реверсированный массив: ${reversedArray.joinToString(", ")}")

}

```



8.

```

fun main() {

    // Создаем массив целых чисел

    val numbers = arrayOf(10, 20, 30, 40, 50, 60, 70, 80, 90)


    // Элемент для поиска

    val target = 50


    // Ищем индекс элемента

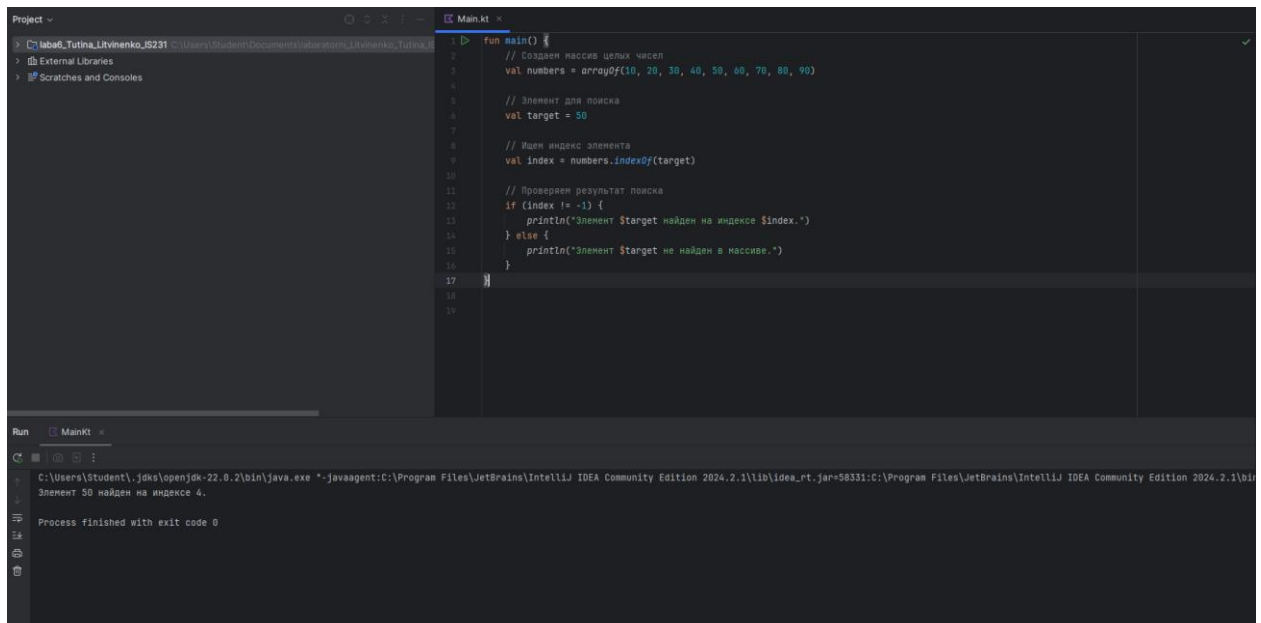
```

```

val index = numbers.indexOf(target)

// Проверяем результат поиска
if (index != -1) {
    println("Элемент $target найден на индексе $index.")
} else {
    println("Элемент $target не найден в массиве.")
}
}

```



9.

```

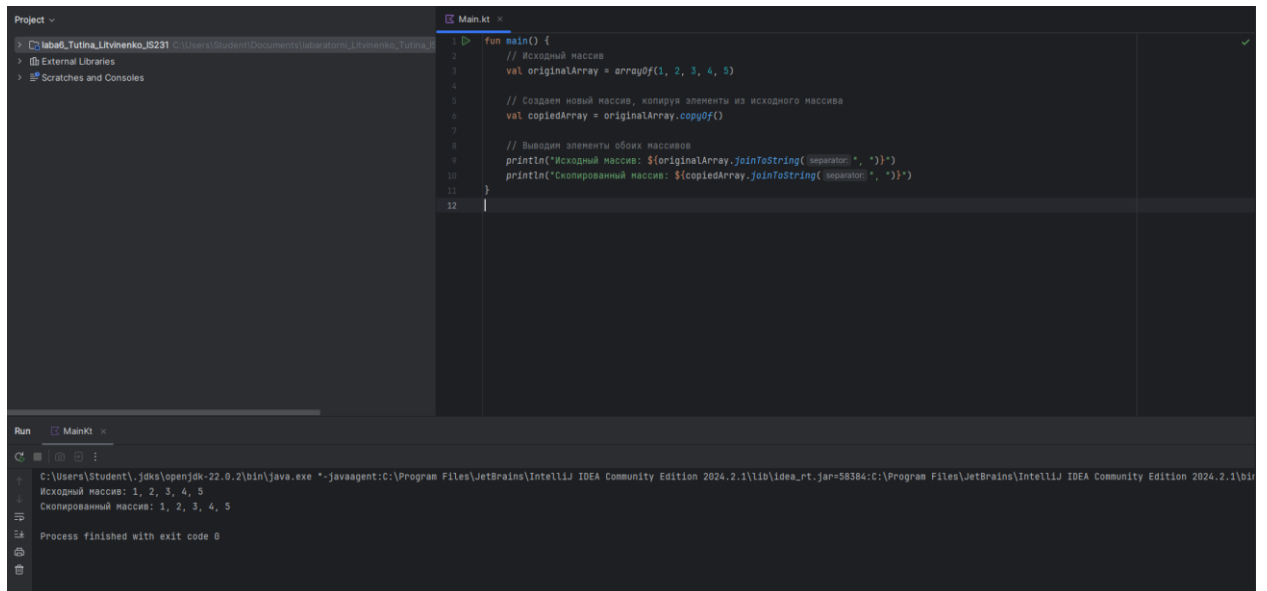
fun main() {
    // Исходный массив
    val originalArray = arrayOf(1, 2, 3, 4, 5)

    // Создаем новый массив, копируя элементы из исходного массива
    val copiedArray = originalArray.copyOf()

    // Выводим элементы обоих массивов
    println("Исходный массив: ${originalArray.joinToString(", ")}")
    println("Скопированный массив: ${copiedArray.joinToString(", ")}")
}

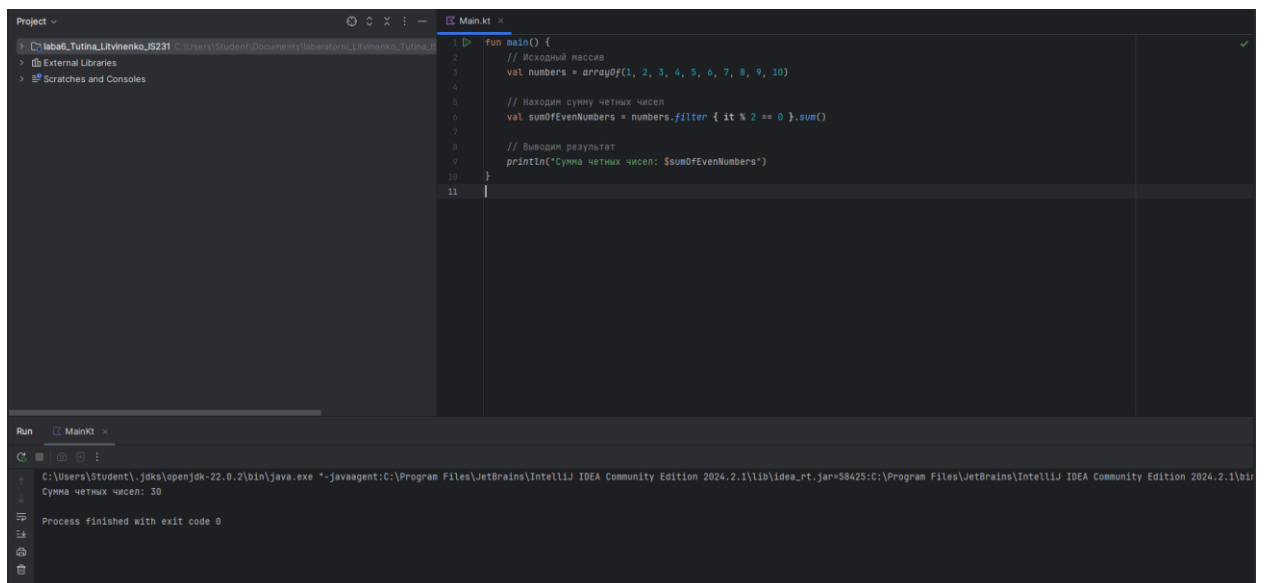
```





10.

```
fun main() {  
    // Исходный массив  
    val numbers = arrayOf(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)  
  
    // Находим сумму четных чисел  
    val sumOfEvenNumbers = numbers.filter { it % 2 == 0 }.sum()  
  
    // Выводим результат  
    println("Сумма четных чисел: $sumOfEvenNumbers")  
}
```



11.

```

fun main() {

    // Исходные массивы

    val array1 = arrayOf(1, 2, 3, 4, 5)

    val array2 = arrayOf(4, 5, 6, 7, 8)


    // Находим пересечение массивов

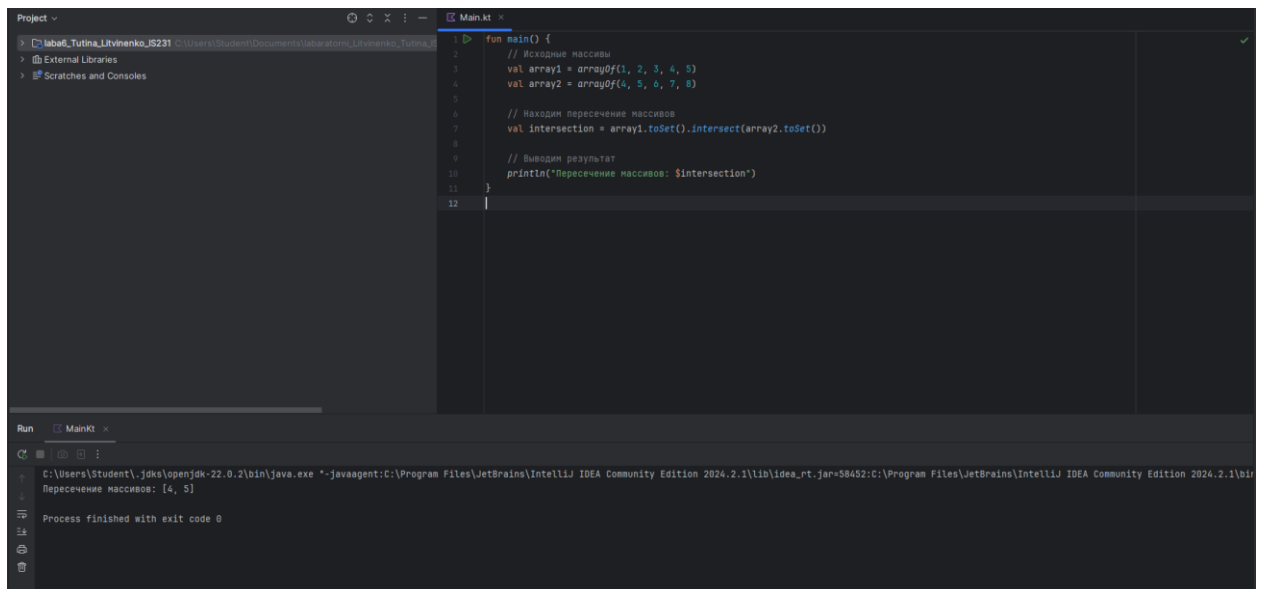
    val intersection = array1.toSet().intersect(array2.toSet())


    // Выводим результат

    println("Пересечение массивов: $intersection")

}

```



12.

```

fun main() {

    // Исходный массив

    val array = arrayOf(1, 2, 3, 4, 5)


    // Индексы элементов, которые нужно поменять местами

    val index1 = 1 // Элемент 2

    val index2 = 3 // Элемент 4


    // Выводим массив до изменения

    println("Массив до изменения: ${array.joinToString(", ")}")
}

```

```

// Меняем местами элементы
swap(array, index1, index2)

// Выводим массив после изменения
println("Массив после изменения: ${array.joinToString(", ")}")
}

// Функция для обмена местами двух элементов в массиве
fun swap(array: Array<Int>, index1: Int, index2: Int) {
    if (index1 < 0 || index1 >= array.size || index2 < 0 || index2 >= array.size) {
        println("Индексы вне диапазона")
        return
    }

    val temp = array[index1]
    array[index1] = array[index2]
    array[index2] = temp
}

```

The screenshot shows an IDE with a Kotlin file named 'Main.kt'. The code implements a swap function and a main function. The main function creates an array [1, 2, 3, 4, 5], prints it, swaps elements at indices 1 and 3, prints the result, and then calls the swap function with the same parameters. The output console shows the array before and after the swap, confirming the swap of elements at indices 1 and 3.

```

1 fun main() {
2     // Исходный массив
3     val array = arrayOf(1, 2, 3, 4, 5)
4
5     // Индексы элементов, которые нужно поменять местами
6     val index1 = 1 // элемент 2
7     val index2 = 3 // элемент 4
8
9     // Выводим массив до изменения
10    println("Массив до изменения: ${array.joinToString(separator: ", ")}")
11
12    // Меняем местами элементы
13    swap(array, index1, index2)
14
15    // Выводим массив после изменения
16    println("Массив после изменения: ${array.joinToString(separator: ", ")}")
17 }
18
19 // Функция для обмена местами двух элементов в массиве
20 fun swap(array: Array<Int>, index1: Int, index2: Int) {
21     if (index1 < 0 || index1 >= array.size || index2 < 0 || index2 >= array.size) {
22         println("Индексы вне диапазона")
23         return
24     }
25
26     val temp = array[index1]
27     array[index1] = array[index2]
28     array[index2] = temp
29 }

```

Run Main.kt

```

C:\Users\Student\jdk\openjdk-22.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.2.1\lib\idea_rt.jar=58493:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.2.1\bin"
Массив до изменения: 1, 2, 3, 4, 5
Массив после изменения: 1, 4, 3, 2, 5
Process finished with exit code 0

```

13.

import kotlin.random.Random

```

fun main() {

    // Создаем массив на 20 элементов

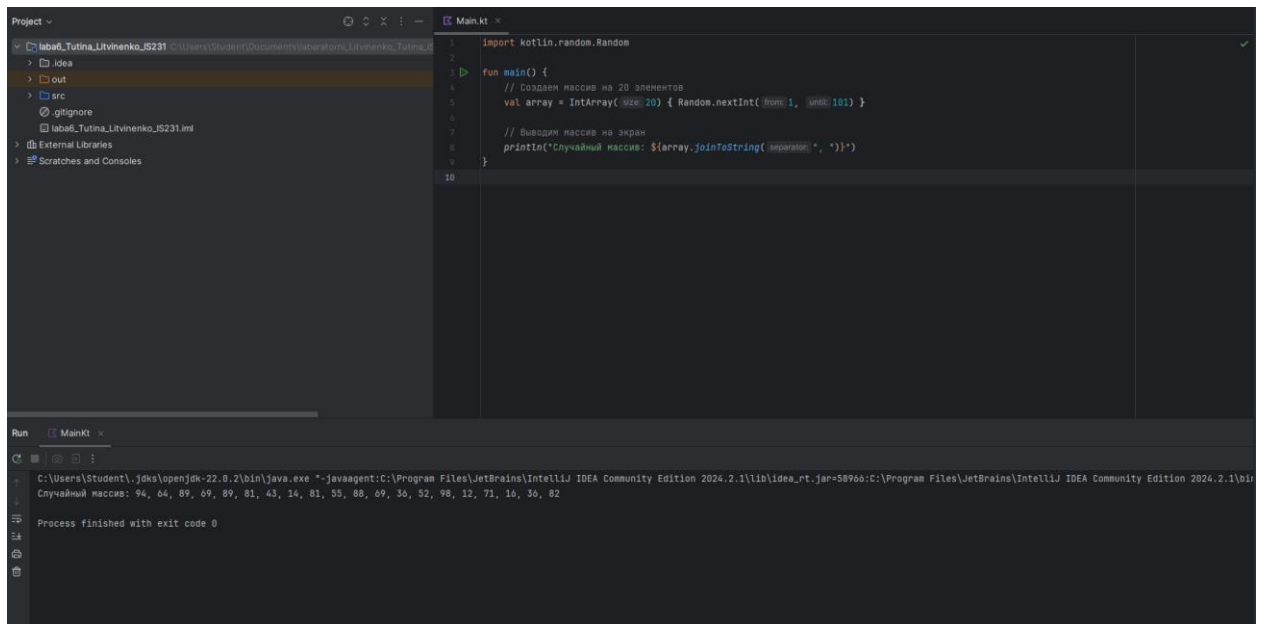
    val array = IntArray(20) { Random.nextInt(1, 101) }


    // Выводим массив на экран

    println("Случайный массив: ${array.joinToString(", ")}")

}

```



14.

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

```

fun main() {

    // Создаем массив на 20 элементов со случайными числами от 1 до 100

    val array = IntArray(20) { Random.nextInt(1, 101) }


    // Выводим оригинальный массив

    println("Оригинальный массив: ${array.joinToString(", ")}")


    // Находим и выводим числа, делящиеся на 3

    val divisibleByThree = array.filter { it % 3 == 0 }

```

// Проверяем, есть ли такие числа и выводим их

```
if (divisibleByThree.isNotEmpty()) {
```

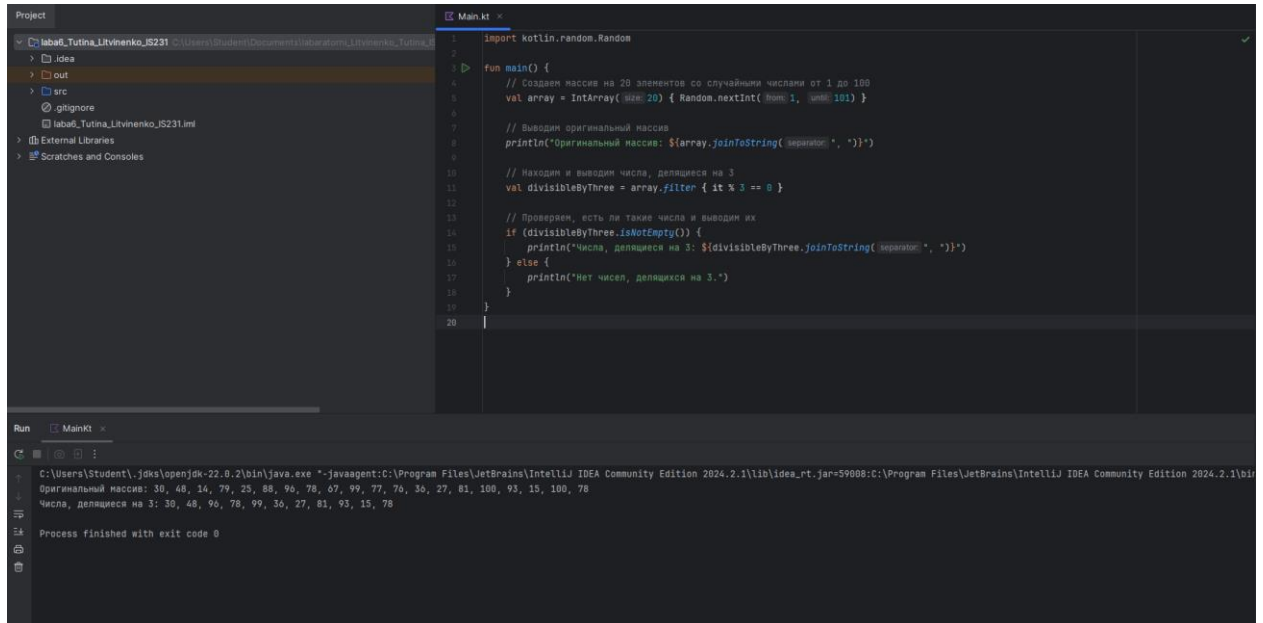
```
    println("Числа, делящиеся на 3: ${divisibleByThree.joinToString(", ")}")
```

```
} else {
```

```
    println("Нет чисел, делящихся на 3.")
```

```
}
```

```
}
```



15.

```
fun main() {
```

```
    // Пример массива
```

```
    val array = arrayOf(1, 2, 3, 2, 1)
```

```
    // Выводим оригинальный массив
```

```
    println("Оригинальный массив: ${array.joinToString(", ")}")
```

```
    // Проверяем, является ли массив палиндромом
```

```
    val isPalindrome = isArrayPalindrome(array)
```

```
    // Выводим результат
```

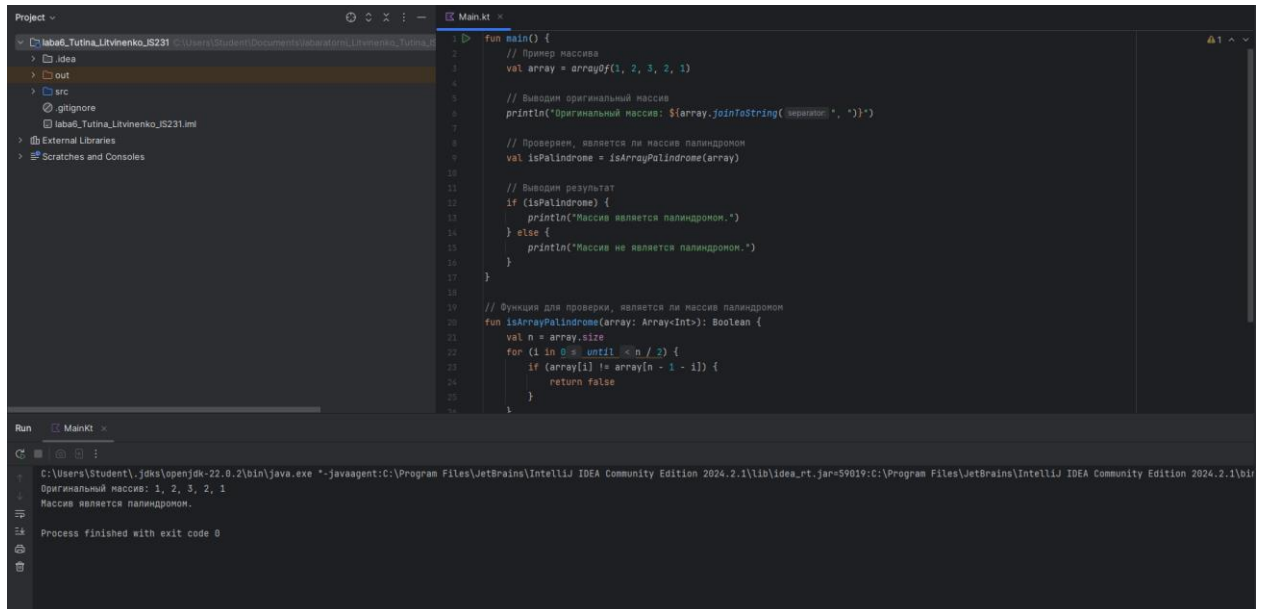
```
    if (isPalindrome) {
```

```

        println("Массив является палиндромом.")
    } else {
        println("Массив не является палиндромом.")
    }
}

// Функция для проверки, является ли массив палиндромом
fun isArrayPalindrome(array: Array<Int>): Boolean {
    val n = array.size
    for (i in 0 until n / 2) {
        if (array[i] != array[n - 1 - i]) {
            return false
        }
    }
    return true
}

```



16.

```

fun main() {
    // Создаем два массива
    val array1 = arrayOf(1, 2, 3)
    val array2 = arrayOf(4, 5, 6)
}

```

```

// Выводим оригинальные массивы

println("Первый массив: ${array1.joinToString(", ")}")

println("Второй массив: ${array2.joinToString(", ")}")

// Конкатенируем массивы

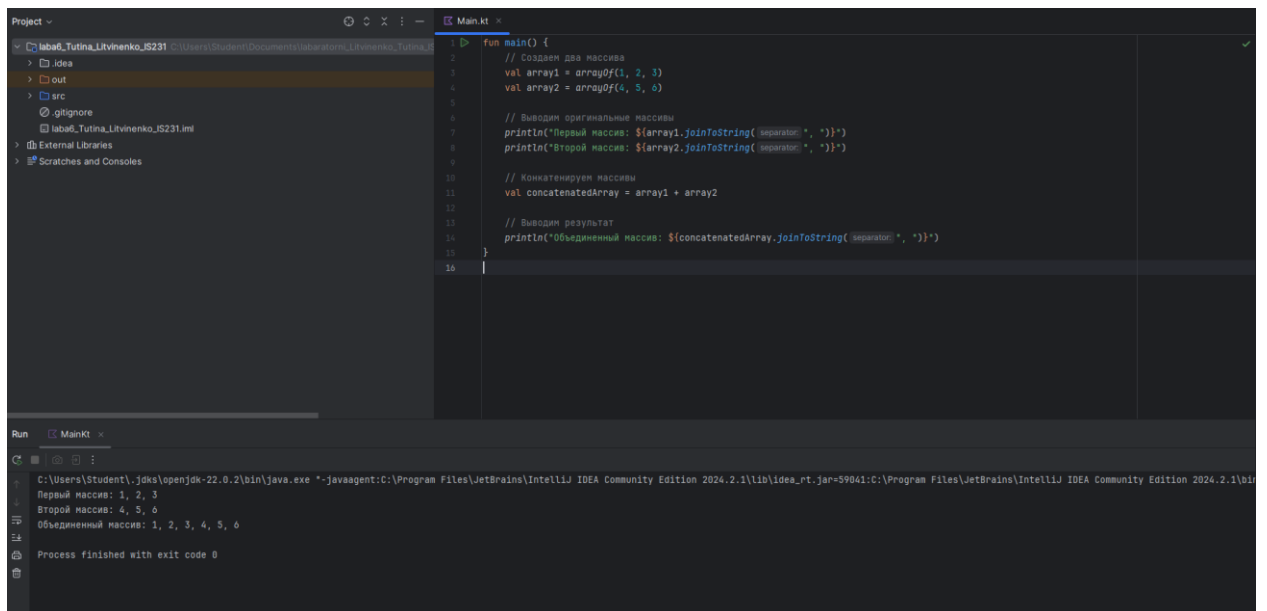
val concatenatedArray = array1 + array2

// Выводим результат

println("Объединенный массив: ${concatenatedArray.joinToString(", ")}")

}

```



17.

```

fun main() {

    // Создаем массив с примерами целых чисел

    val array = arrayOf(1, 2, 3, 4, 5)

    // Инициализируем переменные для суммы и произведения

    var sum = 0

    var product = 1

    // Проходим по каждому элементу массива

```

```

for (element in array) {

    sum += element    // Добавляем элемент к сумме

    product *= element // Умножаем элемент на произведение

}

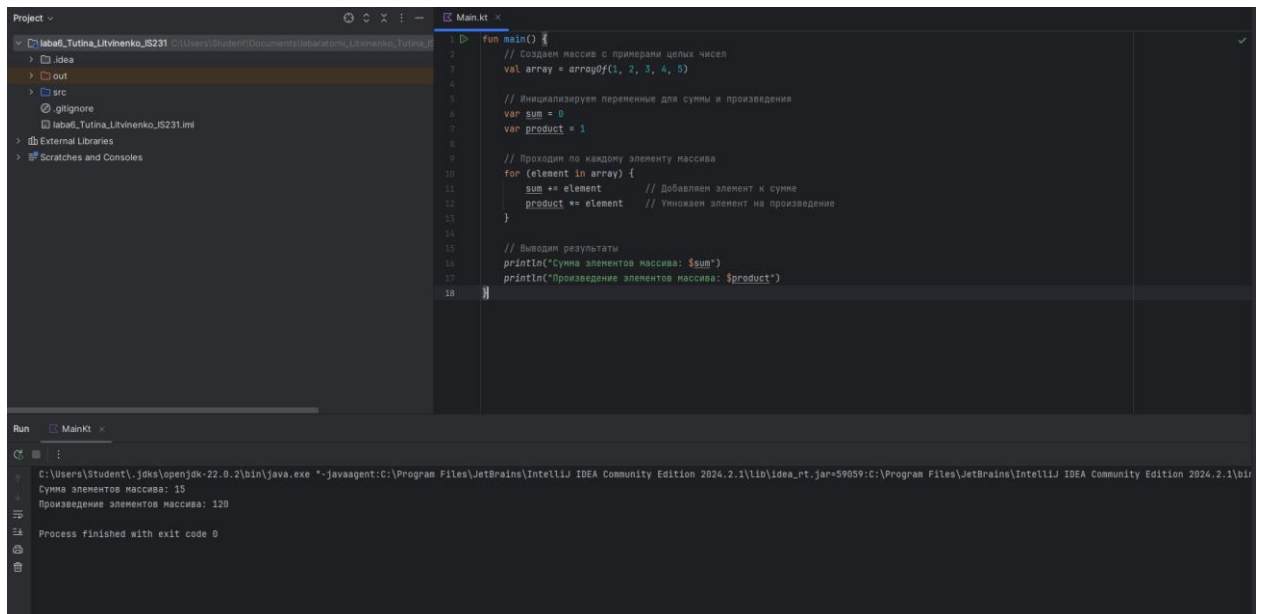
// Выводим результаты

println("Сумма элементов массива: $sum")

println("Произведение элементов массива: $product")

}

```



18.

```

fun main() {

    // Создаем массив с примерами целых чисел

    val array = arrayOf(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20)

    // Определяем размер группы

    val groupSize = 5

    // Проходим по массиву с шагом groupSize

    for (i in array.indices step groupSize) {

        // Создаем подмассив для текущей группы

        val group = array.copyOfRange(i, minOf(i + groupSize, array.size))
    }
}

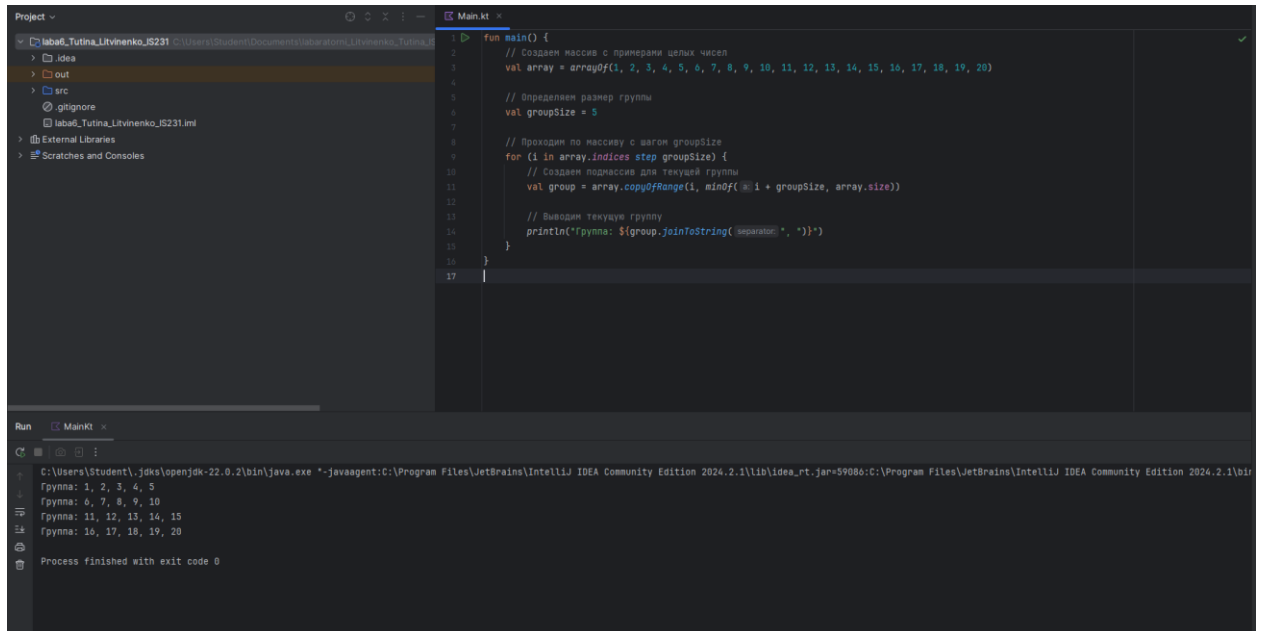
```



```

// Выводим текущую группу
println("Группа: ${group.joinToString(", ")}")
}
}

```



19.

```

fun main() {
    // Два отсортированных массива
    val array1 = arrayOf(1, 3, 5, 7, 9)
    val array2 = arrayOf(2, 4, 6, 8, 10)

    // Слияние массивов
    val mergedArray = mergeSortedArrays(array1, array2)

    // Вывод результата
    println("Слитый отсортированный массив: ${mergedArray.joinToString(", ")}")
}

```

```

fun mergeSortedArrays(array1: Array<Int>, array2: Array<Int>): Array<Int> {
    val merged = mutableListOf<Int>()
    var i = 0

```

```
var j = 0
```

```
// Слияние двух массивов
```

```
while (i < array1.size && j < array2.size) {
```

```
    if (array1[i] < array2[j]) {
```

```
        merged.add(array1[i])
```

```
        i++
```

```
    } else {
```

```
        merged.add(array2[j])
```

```
        j++
```

```
    }
```

```
}
```

```
// Добавляем оставшиеся элементы из первого массива
```

```
while (i < array1.size) {
```

```
    merged.add(array1[i])
```

```
    i++
```

```
}
```

```
// Добавляем оставшиеся элементы из второго массива
```

```
while (j < array2.size) {
```

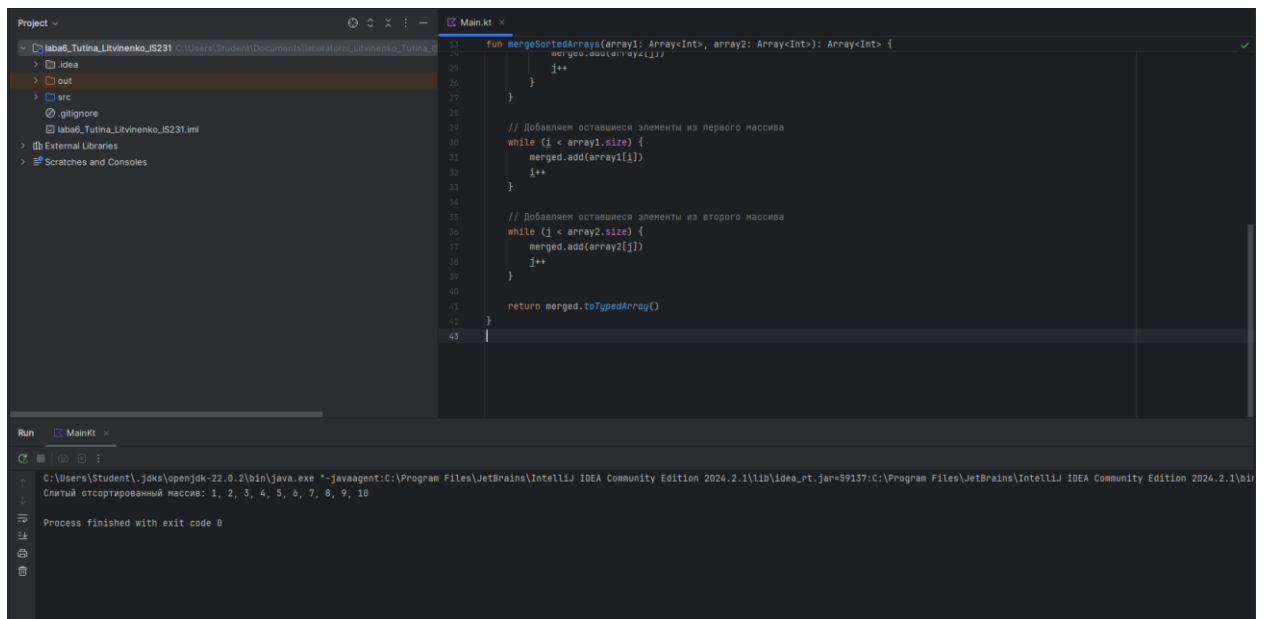
```
    merged.add(array2[j])
```

```
    j++
```

```
}
```

```
return merged.toArray()
```

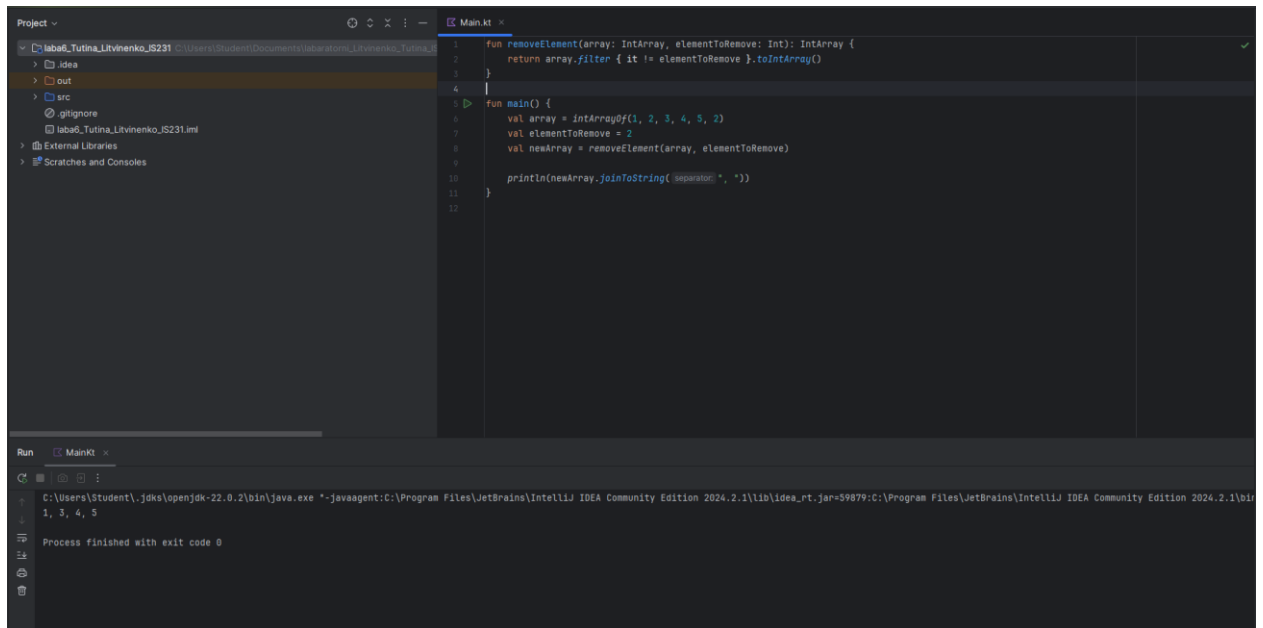
```
}
```



20.

```
fun removeElement(array: IntArray, elementToRemove: Int): IntArray {  
    return array.filter { it != elementToRemove }.toIntArray()  
}
```

```
fun main() {  
    val array = intArrayOf(1, 2, 3, 4, 5, 2)  
    val elementToRemove = 2  
    val newArray = removeElement(array, elementToRemove)  
  
    println(newArray.joinToString(", "))  
}
```



21.

```
fun main() {
```

```
    val originalArray = intArrayOf(1, 2, 3, 4, 5, 6, 7, 8, 9)
```

```
    val elementToRemove = 5
```

```
    val updatedArray = removeElement(originalArray, elementToRemove)
```

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

```
    println("Массив после удаления элемента $elementToRemove:  
${updatedArray.joinToString(", ")})")
```

```
}
```

```
fun removeElement(array: IntArray, element: Int): IntArray {
```

```
    // Преобразуем массив в список для удобства
```

```
    val list = array.toMutableList()
```

```
    // Удаляем элемент
```

```
    list.remove(element)
```

```
    // Преобразуем список обратно в массив
```

```
    return list.toIntArray()
```

}

```
1 fun main() {
2     val originalArray = IntArrayOf(1, 2, 3, 4, 5, 6, 7, 8, 9)
3     val elementToRemove = 5
4
5     val updatedArray = removeElement(originalArray, elementToRemove)
6
7     println("Исходный массив: ${originalArray.joinToString(" ")}")
8     println("Массив после удаления элемента $elementToRemove: ${updatedArray.joinToString(" ")}")
9 }
10
11 fun removeElement(array: IntArray, element: Int): IntArray {
12     // Преобразуем массив в список для удобства
13     val list = array.toList()
14
15     // Удаляем элемент
16     list.remove(element)
17
18     // Преобразуем список обратно в массив
19     return list.toIntArray()
20 }
21
```

Run MainKt x

C:\Users\Student\jdk\openjdk-22.0.2\bin\java.exe -javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.2.1\lib\idea\_rt.jar=59348:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.2.1\bin

Исходный массив: 1, 2, 3, 4, 5, 6, 7, 8, 9

Массив после удаления элемента 5: 1, 2, 3, 4, 6, 7, 8, 9

Process finished with exit code 0

22.

```
fun main() {
```

```
    val array = intArrayOf(10, 20, 4, 45, 99, 99, 45)
```

```
    val secondMax = findSecondMax(array)
```

```
    if (secondMax != null) {
```

```
        println("Второй по величине элемент: $secondMax")
```

```
    } else {
```

```
        println("Второго по величине элемента нет.")
```

```
    }
```

```
}
```

```
fun findSecondMax(array: IntArray): Int? {
```

```
    if (array.size < 2) return null // Если элементов меньше двух, возвращаем null
```

```
    var max = Int.MIN_VALUE
```

```
    var secondMax = Int.MIN_VALUE
```

```
    for (num in array) {
```

```

    if (num > max) {

        secondMax = max // Обновляем второй максимальный

        max = num // Обновляем максимальный

    } else if (num > secondMax && num < max) {

        secondMax = num // Обновляем второй максимальный, если текущий элемент
меньше максимального

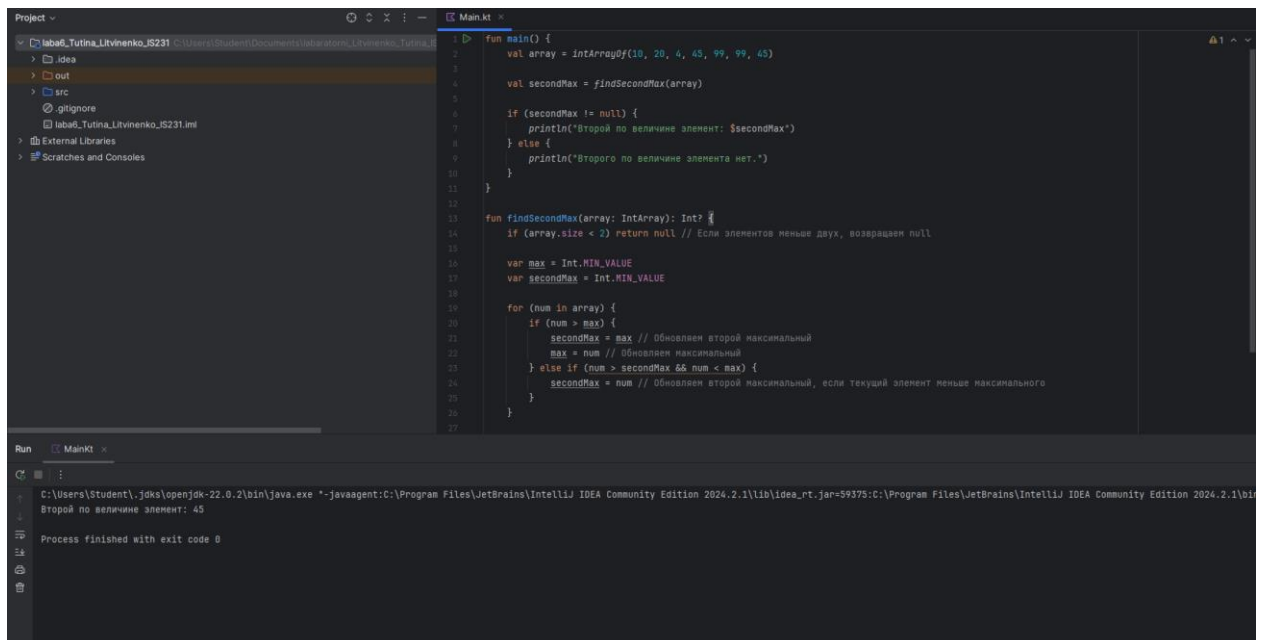
    }

}

return if (secondMax == Int.MIN_VALUE) null else secondMax // Если второй
максимальный не обновился, возвращаем null

}

```



23.

```

fun main() {

    val array1 = intArrayOf(1, 2, 3)

    val array2 = intArrayOf(4, 5)

    val array3 = intArrayOf(6, 7, 8, 9)

    val resultArray = mergeArrays(array1, array2, array3)

    println("Результирующий массив: ${resultArray.joinToString(", ")}")

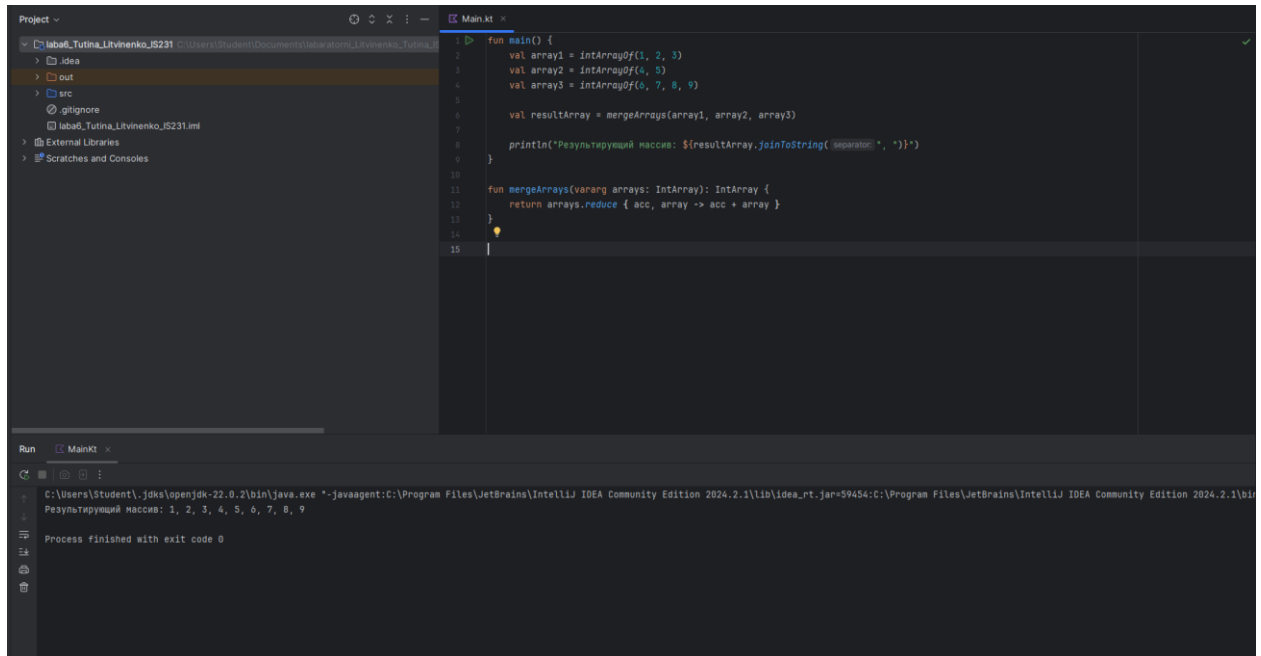
}

```

```

fun mergeArrays(vararg arrays: IntArray): IntArray {
    return arrays.reduce { acc, array -> acc + array }
}

```



24.

```

fun main() {
    // Создаем исходную матрицу
    val matrix = arrayOf(
        intArrayOf(1, 2, 3),
        intArrayOf(4, 5, 6),
        intArrayOf(7, 8, 9)
    )

    println("Исходная матрица:")
    printMatrix(matrix)

    // Транспонируем матрицу
    val transposedMatrix = transposeMatrix(matrix)

    println("Транспонированная матрица:")
    printMatrix(transposedMatrix)
}

```

```
}
```

```
// Функция для транспонирования матрицы
```

```
fun transposeMatrix(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 печати матрицы
```

```
fun printMatrix(matrix: Array<IntArray>) {
```

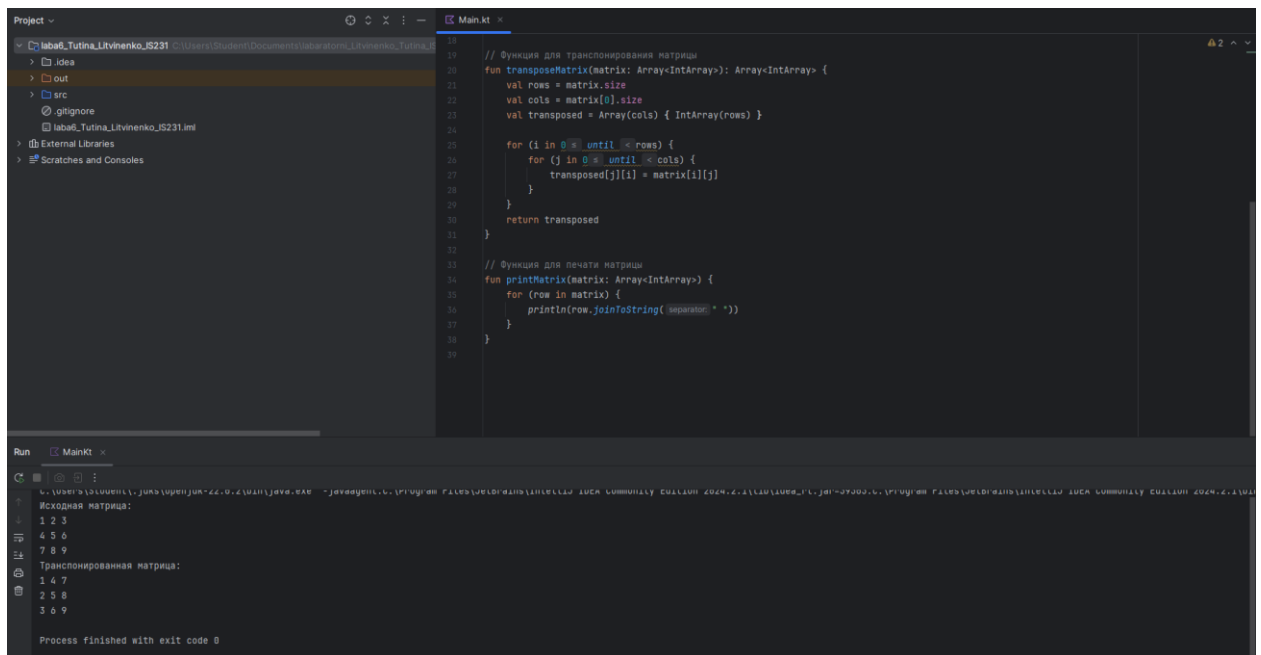
```
    for (row in matrix) {
```

```
        println(row.joinToString(" "))
```

```
    }
```

```
}
```





25.

```

fun main() {

    val array = intArrayOf(1, 3, 5, 7, 9, 11)

    val target = 5

    val found = linearSearch(array, target)

    if (found) {

        println("Элемент $target найден в массиве.")

    } else {

        println("Элемент $target не найден в массиве.")

    }

}

// Функция для линейного поиска
fun linearSearch(array: IntArray, target: Int): Boolean {

    for (element in array) {

        if (element == target) {

            return true // Элемент найден

        }

    }

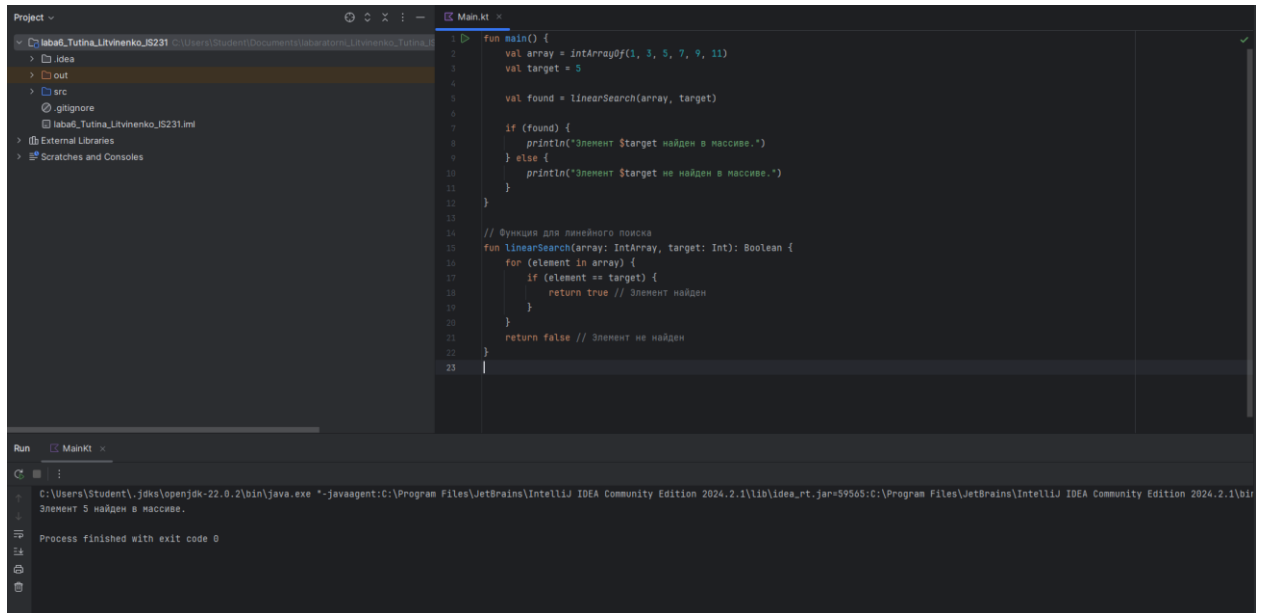
}

```

```

return false // Элемент не найден
}

```



26.

```

fun main() {

    val array = doubleArrayOf(10.0, 20.0, 30.0, 40.0, 50.0)

    val average = calculateAverage(array)

    println("Среднее арифметическое: $average")
}

// Функция для вычисления среднего арифметического
fun calculateAverage(array: DoubleArray): Double {

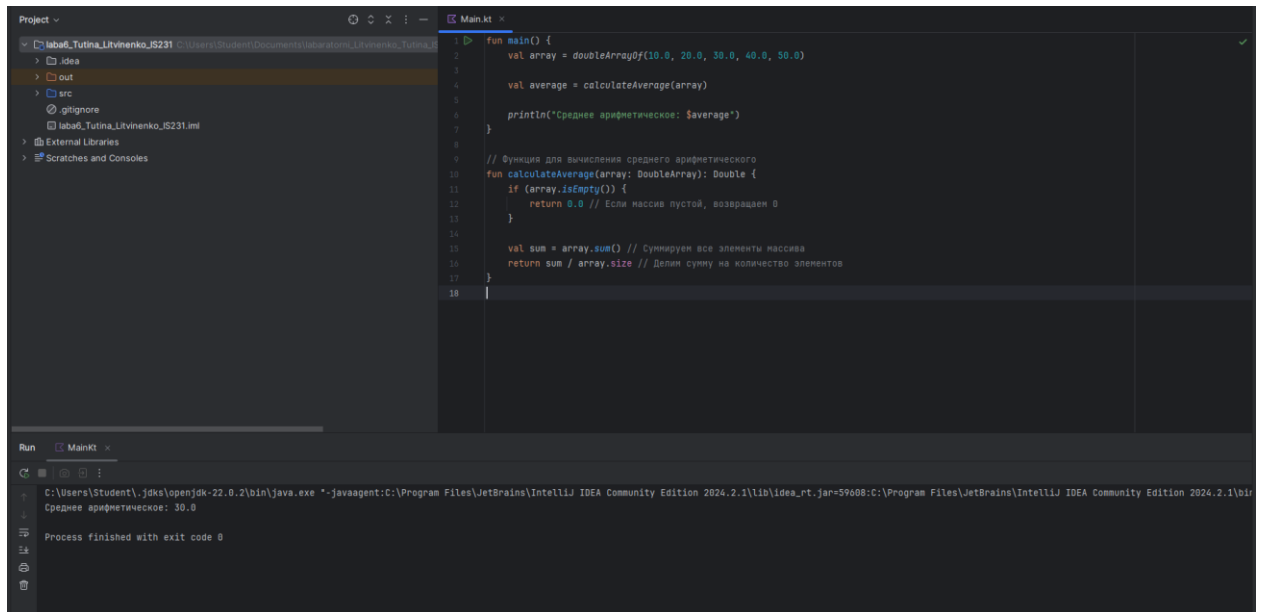
    if (array.isEmpty()) {

        return 0.0 // Если массив пустой, возвращаем 0

    }

    val sum = array.sum() // Суммируем все элементы массива
    return sum / array.size // Делим сумму на количество элементов
}

```



27.

```
fun main() {
```

```
    val array = arrayOf(1, 1, 2, 2, 2, 3, 3, 1, 1, 1, 1)
```

```
    val (element, length) = findMaxSequence(array)
```

```
    println("Максимальная последовательность: элемент = $element, длина = $length")
```

```
}
```

```
// Функция для поиска максимальной последовательности одинаковых элементов
```

```
fun findMaxSequence(array: Array<Int>): Pair<Int, Int> {
```

```
    if (array.isEmpty()) {
```

```
        return Pair(0, 0) // Если массив пустой, возвращаем 0 для элемента и длины
```

```
    }
```

```
    var maxElement = array[0]
```

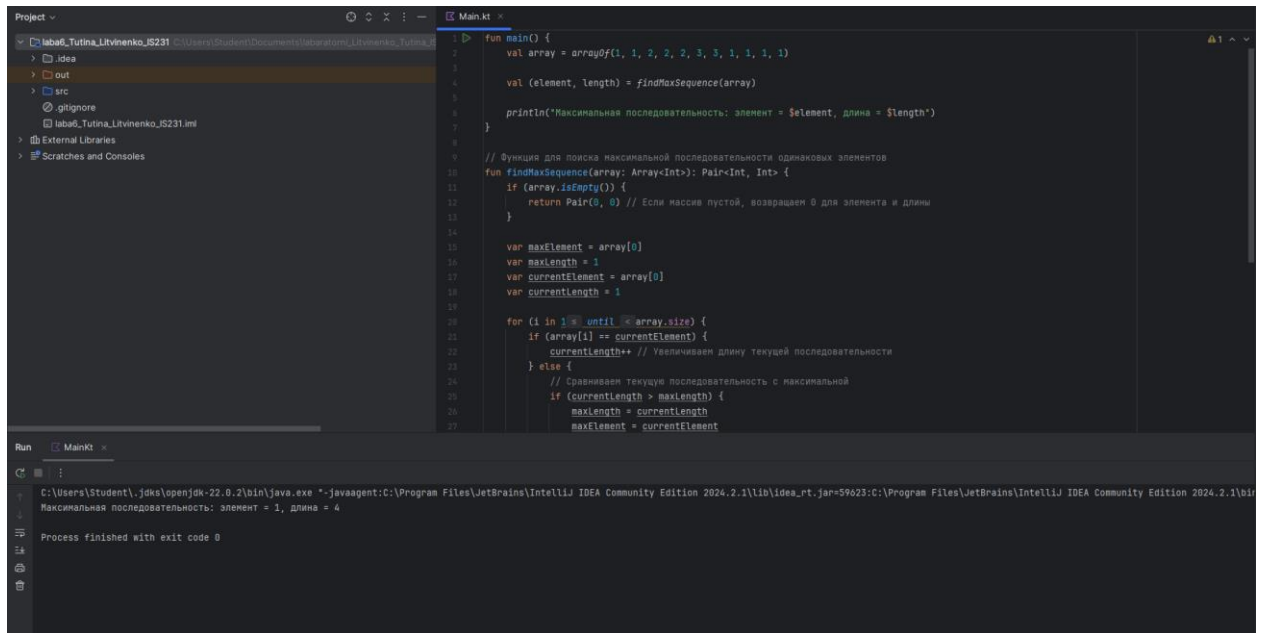
```
    var maxLength = 1
```

```
    var currentElement = array[0]
```

```
    var currentLength = 1
```

```
    for (i in 1 until array.size) {
```

```
    if (array[i] == currentElement) {  
        currentLength++ // Увеличиваем длину текущей последовательности  
    } else {  
        // Сравниваем текущую последовательность с максимальной  
        if (currentLength > maxLength) {  
            maxLength = currentLength  
            maxElement = currentElement  
        }  
        // Сбрасываем текущие значения  
        currentElement = array[i]  
        currentLength = 1  
    }  
}  
  
// Проверяем последнюю последовательность  
if (currentLength > maxLength) {  
    maxLength = currentLength  
    maxElement = currentElement  
}  
  
return Pair(maxElement, maxLength)  
}
```



28.

```
fun main() {
```

```
    // Запрашиваем у пользователя ввод размера массива
```

```
    println("Введите размер массива:")
```

```
    val size = readLine()?.toIntOrNull() ?: return println("Некорректный ввод размера массива.")
```

```
    // Создаем массив указанного размера
```

```
    val array = IntArray(size)
```

```
    // Запрашиваем у пользователя ввод элементов массива
```

```
    println("Введите $size чисел через пробел:")
```

```
    val input = readLine()
```

```
    // Проверяем, что введено значение и разбиваем строку на массив строк
```

```
    input?.split(" ").take(size)?.forEachIndexed { index, s ->
```

```
        array[index] = s.toIntOrNull() ?: return println("Некорректный ввод элемента массива на позиции $index.")
```

```
    }
```

```
    // Выводим массив
```

```
    println("Вы ввели массив: ${array.joinToString(", ")}")
```

[illegible]

```
fun main() {  
    // Запрашиваем у пользователя ввод размера массива  
    println("Введите размер массива:")  
    val size = readLine()?.toIntOrNull() ?: return println("Некорректный ввод размера массива.")  
  
    // Создаем массив указанного размера  
    val array = IntArray(size)  
  
    // Запрашиваем у пользователя ввод элементов массива  
    println("Введите $size чисел через пробел:")  
    val input = readLine()  
  
    // Проверяем, что введено значение и разбиваем строку на массив строк  
    input?.split(" ").take(size)?.forEachIndexed { index, s ->  
        array[index] = s.toIntOrNull() ?: return println("Некорректный ввод элемента массива на позиции $index.")  
    }  
}
```



// Создаем массив из 100 целых чисел

```
val numbers = IntArray(100) { (1..100).random() } // Заполняем массив случайными числами от 1 до 100
```

// Разделяем массив на 10 групп по 10 элементов

```
val groups = Array(10) { IntArray(10) }
```

```
for (i in numbers.indices) {
```

```
    val groupIndex = i / 10 // Определяем индекс группы
```

```
    val elementIndex = i % 10 // Определяем индекс элемента в группе
```

```
    groups[groupIndex][elementIndex] = numbers[i] // Заполняем группу
```

```
}
```

// Выводим результаты

```
for (i in groups.indices) {
```

```
    println("Группа ${i + 1}: ${groups[i].joinToString(", ")}")
```

```
}
```

```
}
```

The screenshot shows an IDE with a project named 'lab06\_Tutina\_Litvinenko\_J5231'. The main file 'Main.kt' contains the following Kotlin code:

```
1 fun main() {  
2     // Создаем массив из 100 целых чисел  
3     val numbers = IntArray(100) { (1..100).random() } // Заполняем массив случайными числами от 1 до 100  
4  
5     // Разделяем массив на 10 групп по 10 элементов  
6     val groups = Array(10) { IntArray(10) }  
7  
8     for (i in numbers.indices) {  
9         val groupIndex = i / 10 // Определяем индекс группы  
10        val elementIndex = i % 10 // Определяем индекс элемента в группе  
11        groups[groupIndex][elementIndex] = numbers[i] // Заполняем группу  
12    }  
13  
14    // Выводим результаты  
15    for (i in groups.indices) {  
16        println("Группа ${i + 1}: ${groups[i].joinToString(", ")}")  
17    }  
18 }  
19
```

The Run console at the bottom displays the output of the program, showing 10 groups of 10 numbers each:

```
Группа 1: 97, 42, 47, 94, 10, 07, 30, 77, 00, 30  
Группа 2: 32, 55, 0, 84, 32, 49, 58, 15, 81, 34  
Группа 3: 46, 18, 64, 21, 3, 31, 98, 73, 15, 17  
Группа 4: 60, 79, 69, 39, 94, 41, 36, 14, 92, 55  
Группа 5: 97, 35, 71, 95, 40, 51, 90, 50, 40, 100  
Группа 6: 4, 48, 67, 1, 39, 69, 21, 65, 60, 92  
Группа 7: 41, 79, 59, 93, 91, 41, 98, 13, 32, 42  
Группа 8: 78, 88, 38, 56, 89, 30, 85, 94, 40, 23  
Группа 9: 17, 91, 19, 88, 49, 55, 38, 65, 50, 4  
Группа 10: 17, 91, 19, 88, 49, 55, 38, 65, 50, 4
```

Process finished with exit code 0