

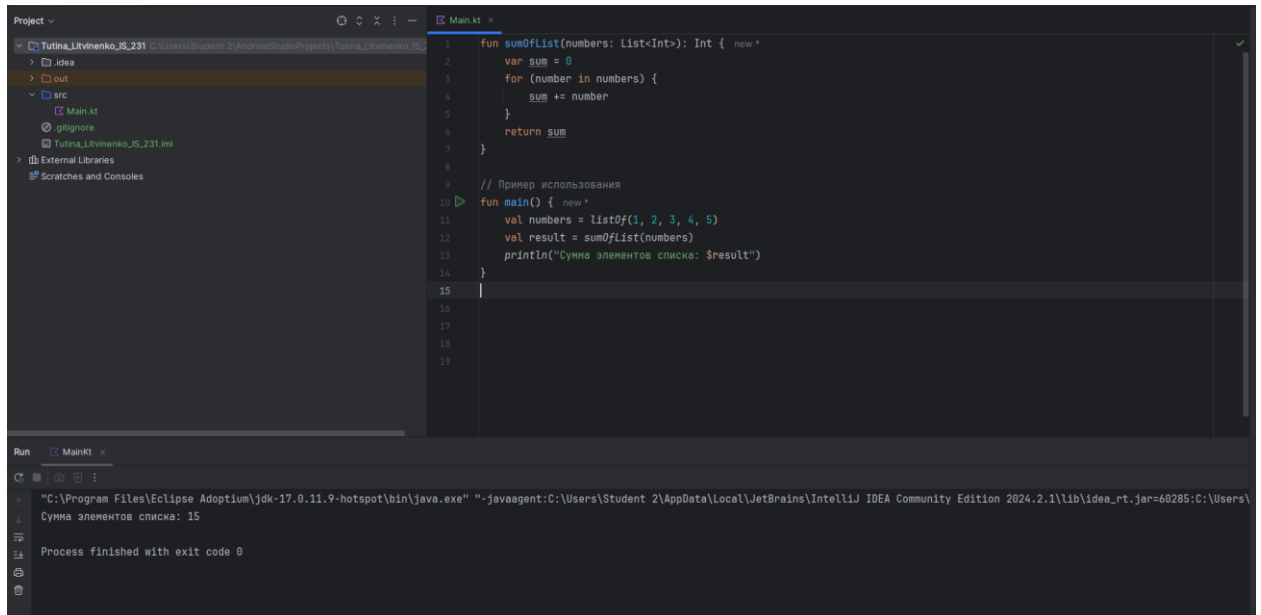
## Практическая работа №8

1.

```
fun sumOfList(numbers: List<Int>): Int {  
    var sum = 0  
  
    for (number in numbers) {  
        sum += number  
    }  
  
    return sum  
}
```

// Пример использования

```
fun main() {  
    val numbers = listOf(1, 2, 3, 4, 5)  
    val result = sumOfList(numbers)  
    println("Сумма элементов списка: $result")  
}
```



2.

```
fun разностьМаксМини(числа: List<Int>): Int {  
    if (числа.isEmpty()) {  
        return 0 // Или можно выбросить исключение, если пустой список недопустим  
    }  
}
```

```

var максимум = числа[0]
var минимум = числа[0]

for (число in числа) {
    if (число > максимум) {
        максимум = число
    }
    if (число < минимум) {
        минимум = число
    }
}

return максимум - минимум
}

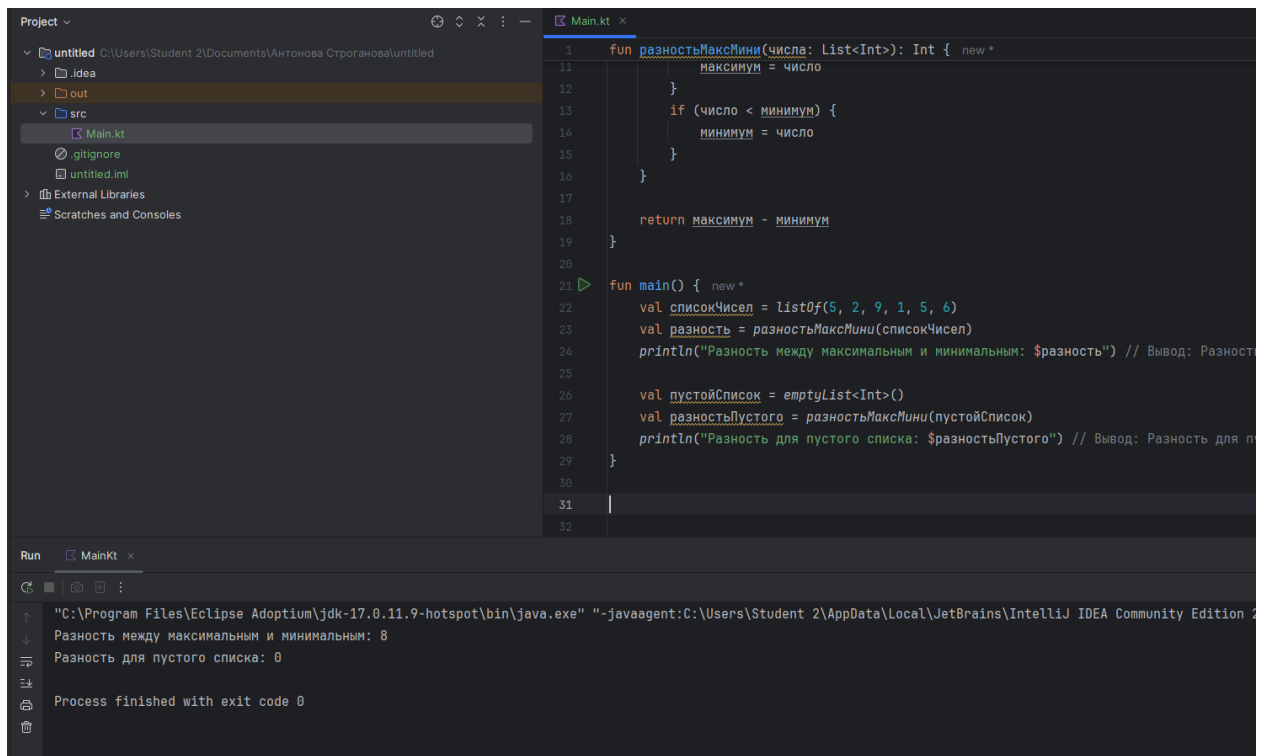
fun main() {
    val списокЧисел = listOf(5, 2, 9, 1, 5, 6)
    val разность = разностьМаксМини(списокЧисел)

    println("Разность между максимальным и минимальным: $разность") // Вывод: Разность
    между максимальным и минимальным: 8

    val пустойСписок = emptyList<Int>()
    val разностьПустого = разностьМаксМини(пустойСписок)

    println("Разность для пустого списка: $разностьПустого") // Вывод: Разность для
    пустого списка: 0
}

```



3.

```

fun объединитьСписки(list1: List<Int>, list2: List<Int>): List<Int> {
    return list1 + list2
}
fun main() {
    val список1 = listOf(1, 2, 3)
    val список2 = listOf(4, 5, 6)

    val объединенныйСписок = объединитьСписки(список1, список2)

    println(объединенныйСписок)
}

```

4.

```

fun profitableGamble(prob: Double, prize: Double, pay: Double): Boolean {

    return prob * prize > pay

}

```

// Example usage:

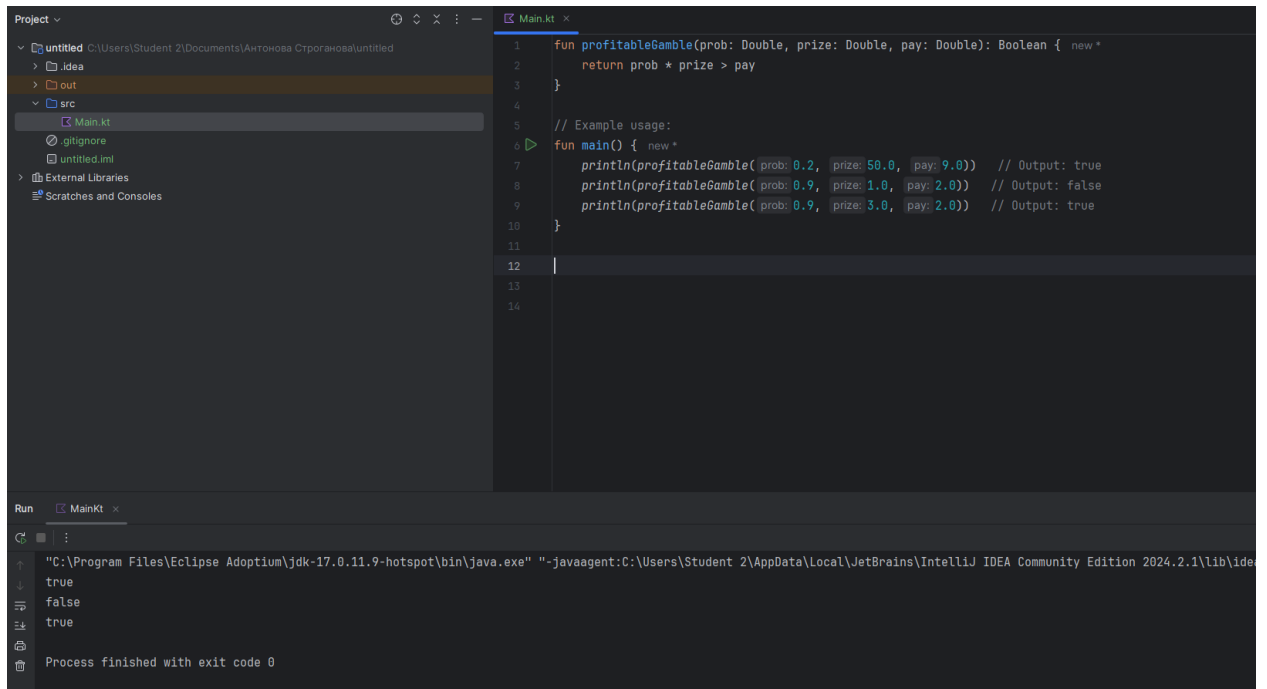
```

fun main() {

    println(profitableGamble(0.2, 50.0, 9.0)) // Output: true
    println(profitableGamble(0.9, 1.0, 2.0)) // Output: false
    println(profitableGamble(0.9, 3.0, 2.0)) // Output: true

}

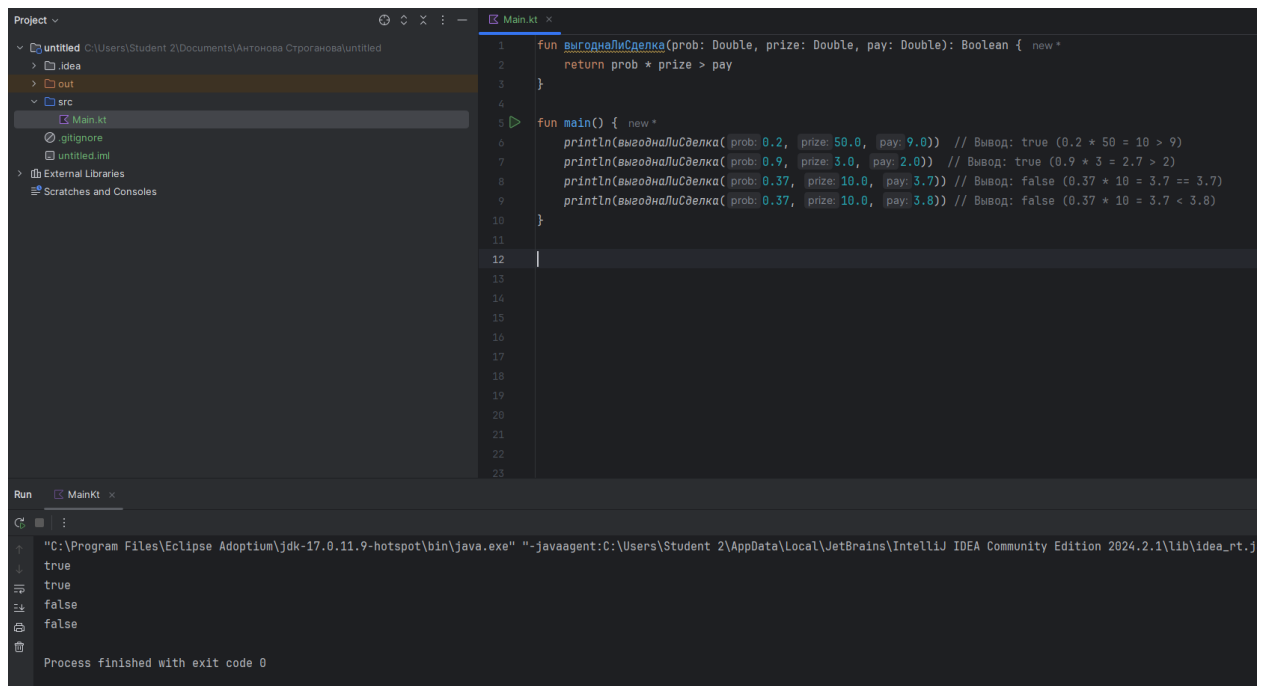
```



5.

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

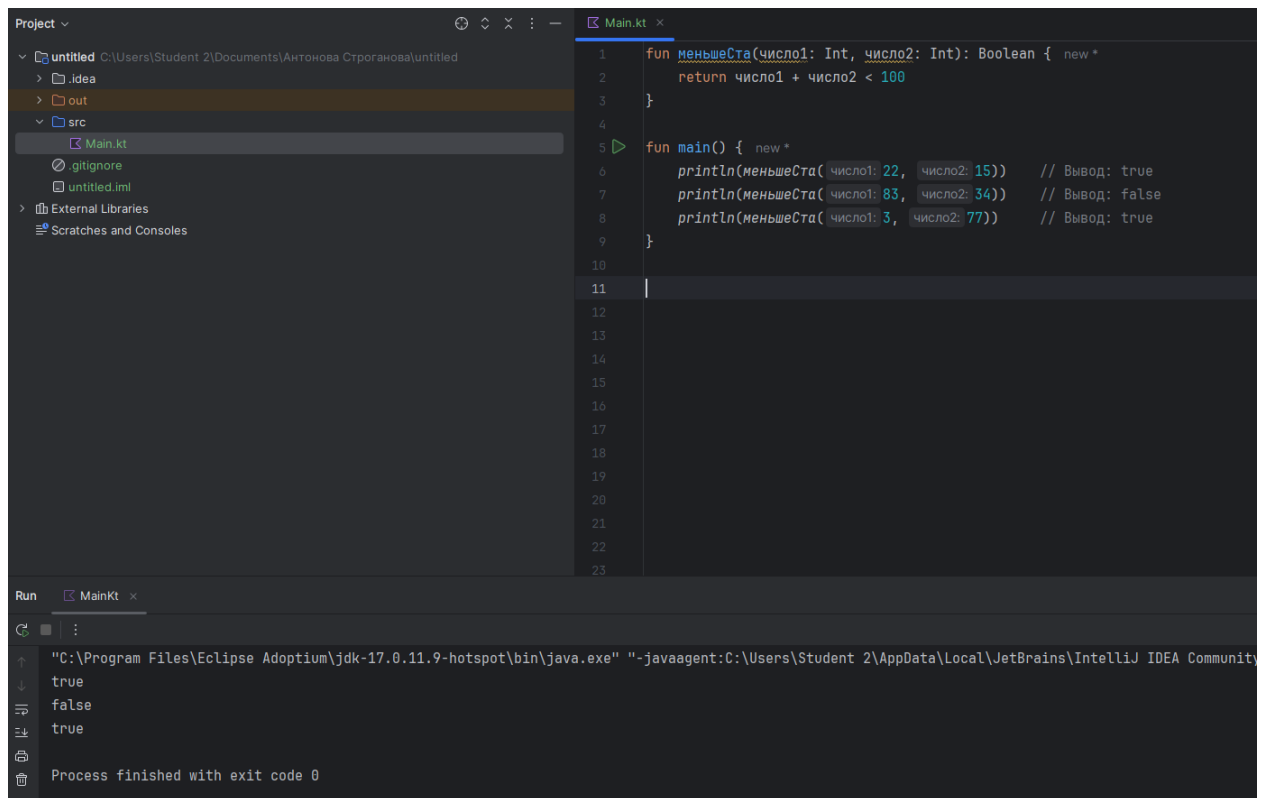
```
fun main() {  
    println(выгодноЛиСделка(0.2, 50.0, 9.0)) // Вывод: true (0.2 * 50 = 10 > 9)  
    println(выгодноЛиСделка(0.9, 3.0, 2.0)) // Вывод: true (0.9 * 3 = 2.7 > 2)  
    println(выгодноЛиСделка(0.37, 10.0, 3.7)) // Вывод: false (0.37 * 10 = 3.7 == 3.7)  
    println(выгодноЛиСделка(0.37, 10.0, 3.8)) // Вывод: false (0.37 * 10 = 3.7 < 3.8)  
}
```



6.

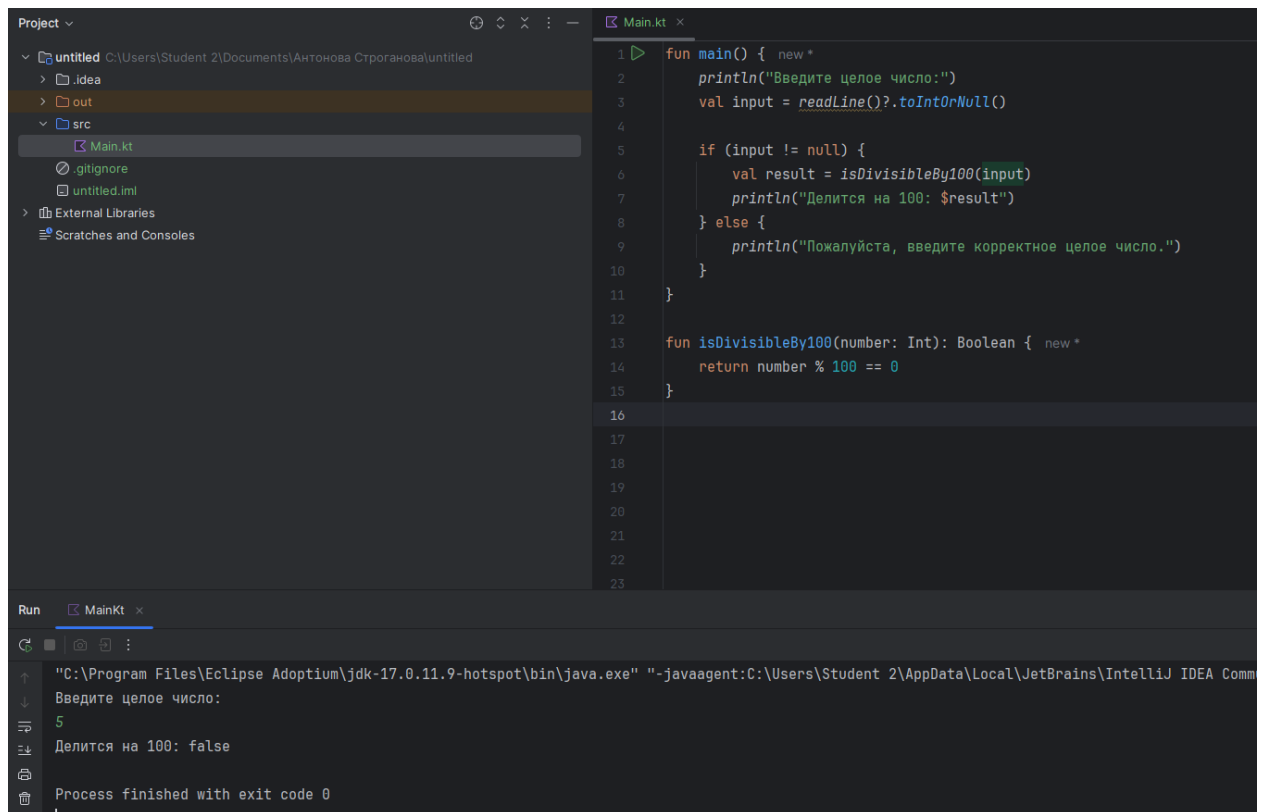
```
fun меньшеСта(число1: Int, число2: Int): Boolean {  
    return число1 + число2 < 100  
}
```

```
fun main() {  
    println(меньшеСта(22, 15)) // Вывод: true  
    println(меньшеСта(83, 34)) // Вывод: false  
    println(меньшеСта(3, 77)) // Вывод: true  
}
```



7.

```
fun main() {  
    println("Введите целое число:")  
    val input = readLine()?.toIntOrNull()  
  
    if (input != null) {  
        val result = isDivisibleBy100(input)  
        println("Делится на 100: $result")  
    } else {  
        println("Пожалуйста, введите корректное целое число.")  
    }  
}  
  
fun isDivisibleBy100(number: Int): Boolean {  
    return number % 100 == 0  
}
```

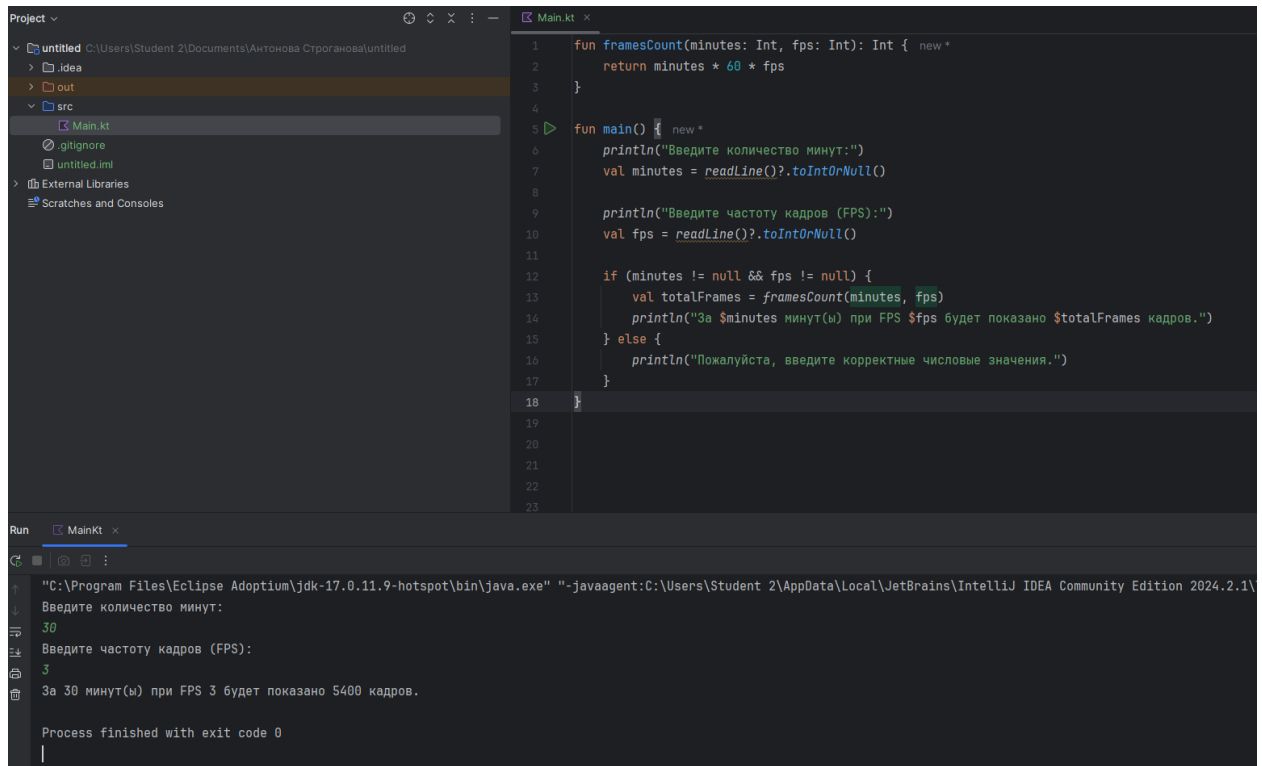


8.

```
fun framesCount(minutes: Int, fps: Int): Int {  
    return minutes * 60 * fps  
}
```

```
fun main() {  
    println("Введите количество минут:")  
    val minutes = readLine()?.toIntOrNull()  
  
    println("Введите частоту кадров (FPS):")  
    val fps = readLine()?.toIntOrNull()  
  
    if (minutes != null && fps != null) {  
        val totalFrames = framesCount(minutes, fps)  
        println("За $minutes минут(ы) при FPS $fps будет показано $totalFrames кадров.")  
    } else {  
        println("Пожалуйста, введите корректные числовые значения.")  
    }  
}
```

}



```
Project: untitled C:\Users\Student 2\Documents\Антонова Стрoганова\untitled
src
  Main.kt
  .gitignore
  untitled.iml
External Libraries
Scratches and Consoles

1 fun framesCount(minutes: Int, fps: Int): Int { new *
2   return minutes * 60 * fps
3 }
4
5 fun main() { new *
6   println("Введите количество минут:")
7   val minutes = readLine()?.toIntOrNull()
8
9   println("Введите частоту кадров (FPS):")
10  val fps = readLine()?.toIntOrNull()
11
12  if (minutes != null && fps != null) {
13    val totalFrames = framesCount(minutes, fps)
14    println("За $minutes минут(ы) при FPS $fps будет показано $totalFrames кадров.")
15  } else {
16    println("Пожалуйста, введите корректные числовые значения.")
17  }
18 }
19
20
21
22
23

Run: MainKt
"C:\Program Files\Eclipse Adoptium\jdk-17.0.11.9-hotspot\bin\java.exe" "-javaagent:C:\Users\Student 2\AppData\Local\JetBrains\IntelliJ IDEA Community Edition 2024.2.1\
Введите количество минут:
30
Введите частоту кадров (FPS):
3
За 30 минут(ы) при FPS 3 будет показано 5400 кадров.
Process finished with exit code 0
```

9.

```
fun isPowerEqual(n: Int, k: Int): Boolean {
    return if (k <= 0) {
        false
    } else {
        val result = Math.pow(k.toDouble(), k.toDouble()).toInt()
        result == n
    }
}
```

```
fun main() {
    println("Введите значение n:")
    val n = readLine()?.toIntOrNull()

    println("Введите значение k:")
    val k = readLine()?.toIntOrNull()

    if (n != null && k != null) {
```

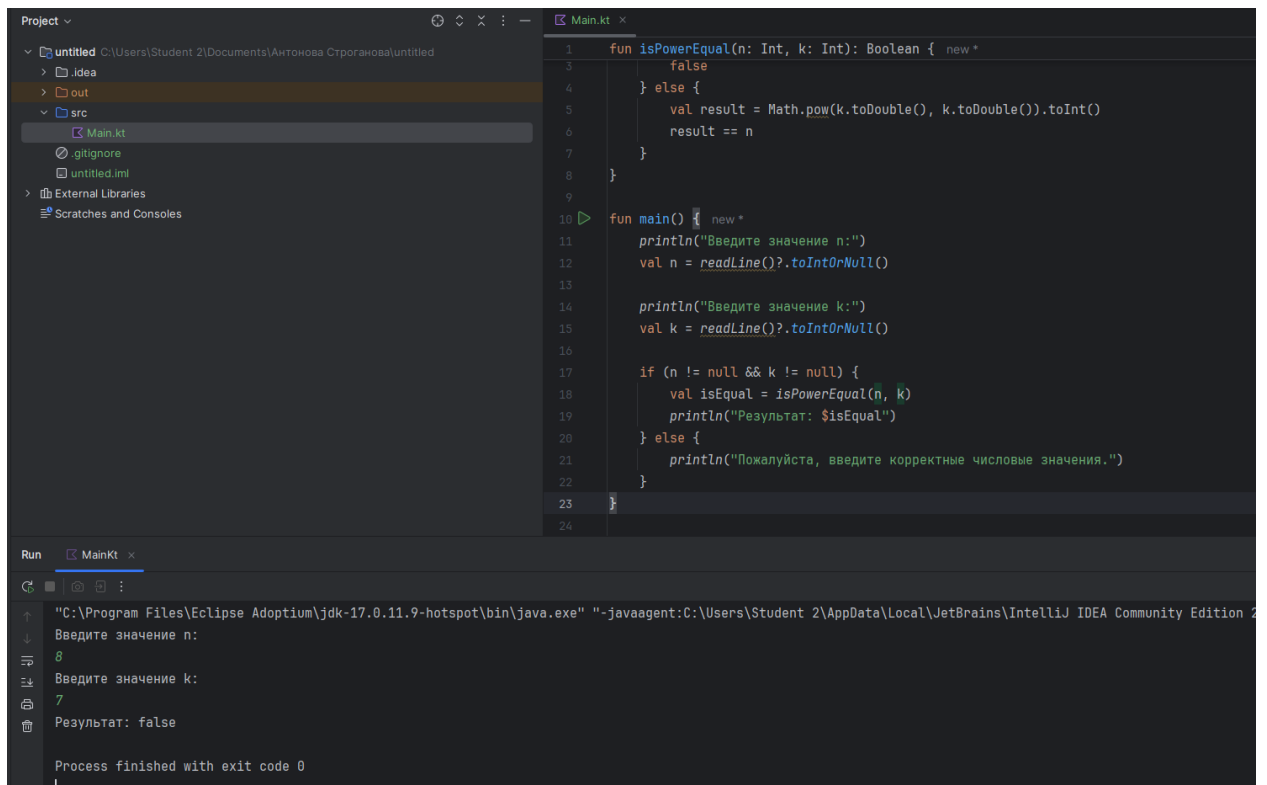


```

    val isEqual = isPowerEqual(n, k)

    println("Результат: $isEqual")
} else {
    println("Пожалуйста, введите корректные числовые значения.")
}
}
}

```



10.

```

fun repeatString(txt: String, n: Int): String {
    return if (n <= 0) {
        ""
    } else {
        txt + repeatString(txt, n - 1)
    }
}

```

```

fun main() {
    println("Введите строку для повторения:")
    val txt = readLine() ?: ""
}

```

```

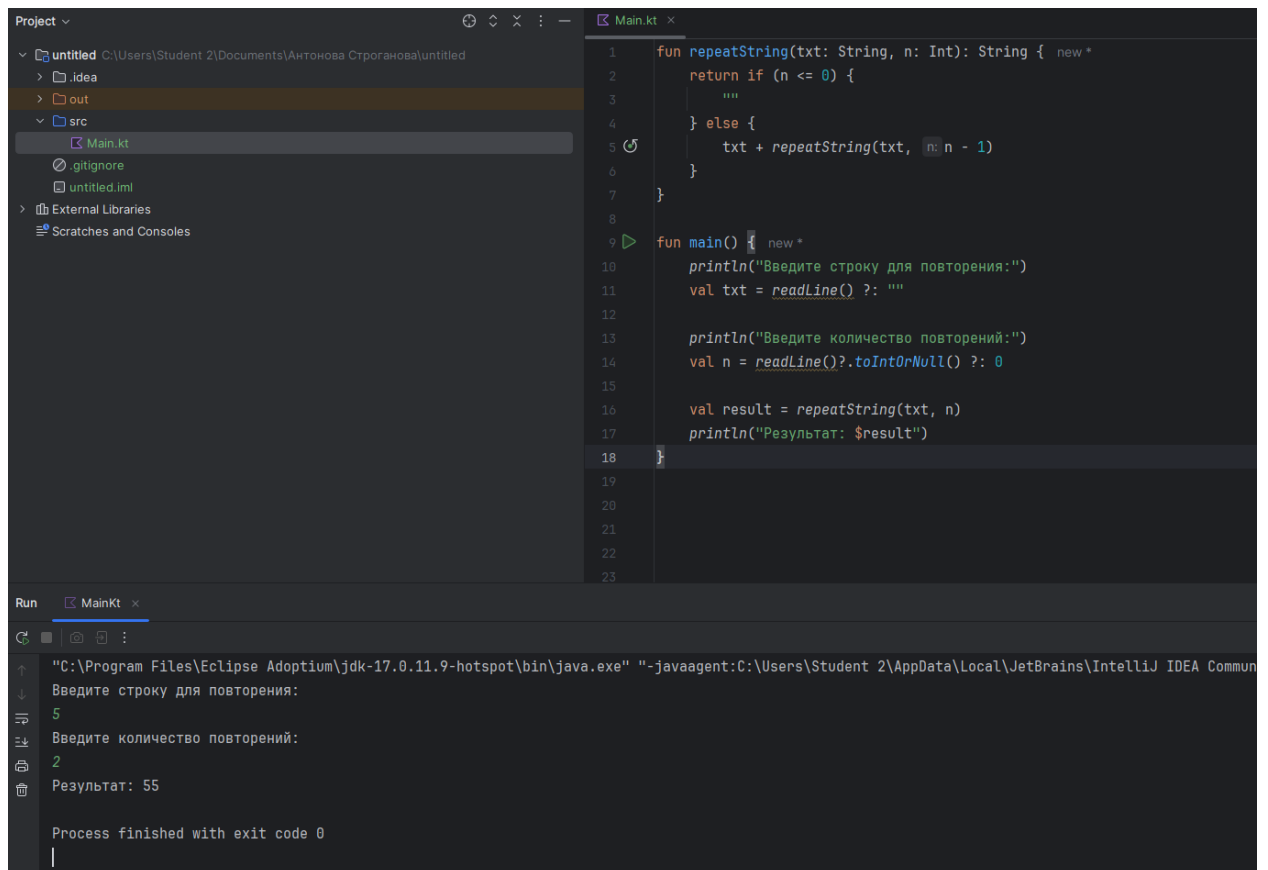
println("Введите количество повторений:")

val n = readLine()?.toIntOrNull() ?: 0

val result = repeatString(txt, n)

println("Результат: $result")
}

```



11.

```
import javax.script.ScriptEngineManager
```

```

fun evaluateEquation(equation: String): Double? {
    val engine = ScriptEngineManager().getEngineByName("JavaScript")
    return try {
        engine.eval(equation) as? Double
    } catch (e: Exception) {
        null
    }
}

```

```

fun main() {
    println("Введите уравнение (например, 1+1):")
    val equation = readLine() ?: ""

    val result = evaluateEquation(equation)
    if (result != null) {
        println("Результат: $result")
    } else {
        println("Ошибка в уравнении.")
    }
}

```

The screenshot shows an IDE with a project named 'untitled'. The source file 'Main.kt' is open, displaying the following Kotlin code:

```

1  import javax.script.ScriptEngineManager
2
3  fun evaluateEquation(equation: String): Double? { new *
4      val engine = ScriptEngineManager().getEngineByName( shortName: "JavaScript")
5      return try {
6          engine.eval(equation) as? Double
7      } catch (e: Exception) {
8          null
9      }
10 }
11
12 fun main() { new *
13     println("Введите уравнение (например, 1+1):")
14     val equation = readLine() ?: ""
15
16     val result = evaluateEquation(equation)
17     if (result != null) {
18         println("Результат: $result")
19     } else {
20         println("Ошибка в уравнении.")
21     }
22 }
23

```

The Run console at the bottom shows the execution of the program:

```

Run MainKt x
"C:\Program Files\Eclipse Adoptium\jdk-17.0.11.9-hotspot\bin\java.exe" "-javaagent:C:\Users\Student 2\AppData\Local\JetBrains\IntelliJ IDEA Community Edition 2024
Введите уравнение (например, 1+1):
2+1
Ошибка в уравнении.
Process finished with exit code 0

```

12.

```

fun generateGoogle(number: Int): String {
    return "G" + "o".repeat(number) + "gle"
}

```

```

fun main() {
    println("Введите количество букв о в слове Google:")
}

```

```
val input = readLine()?.toIntOrNull()
```

```
if (input != null && input >= 0) {
```

```
    val result = generateGoogle(input)
```

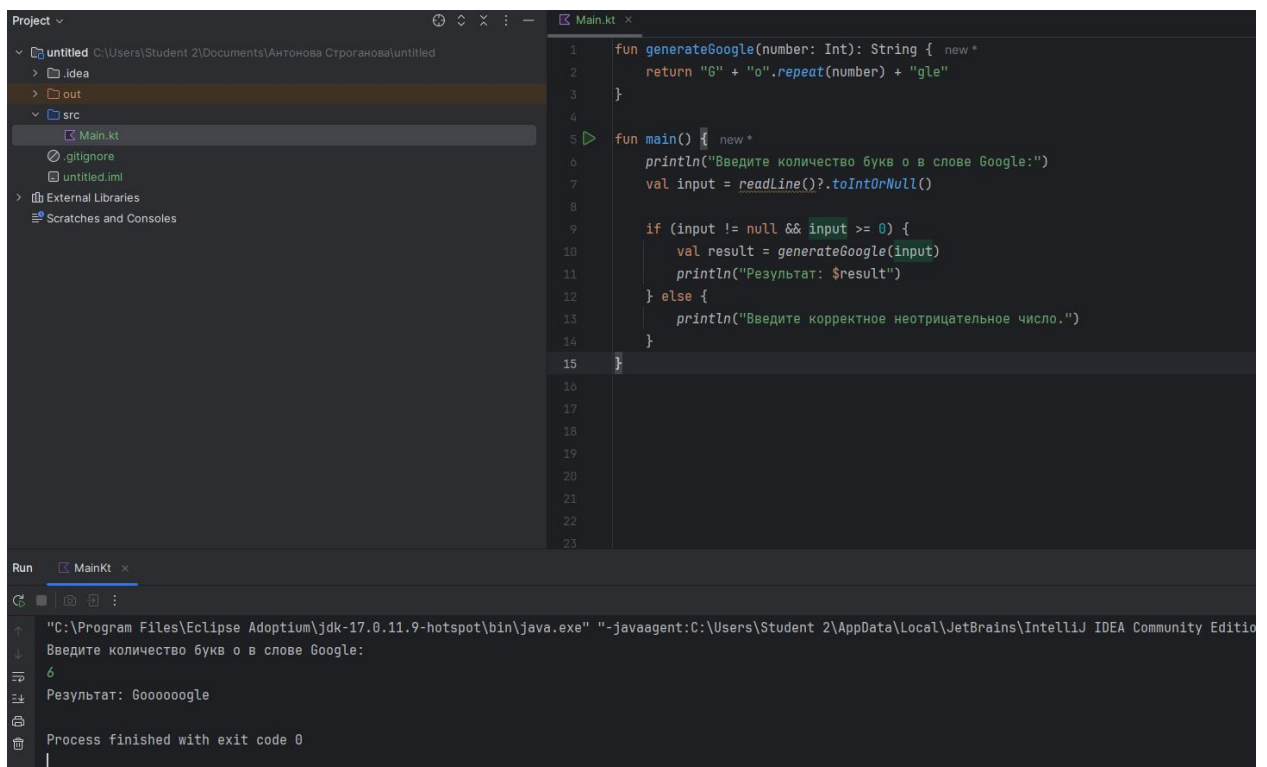
```
    println("Результат: $result")
```

```
} else {
```

```
    println("Введите корректное неотрицательное число.")
```

```
}
```

```
}
```



13.

```
fun greet() {
```

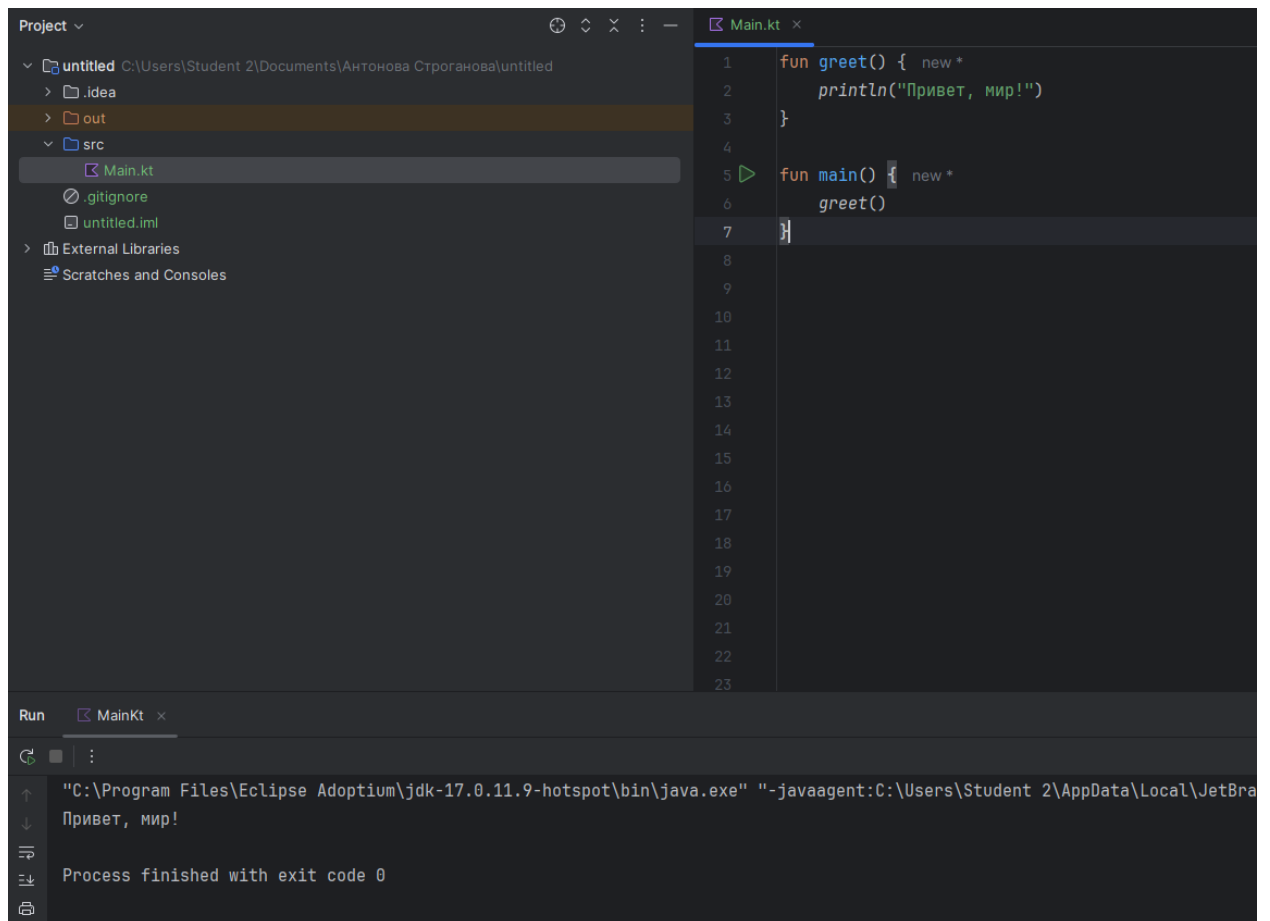
```
    println("Привет, мир!")
```

```
}
```

```
fun main() {
```

```
    greet()
```

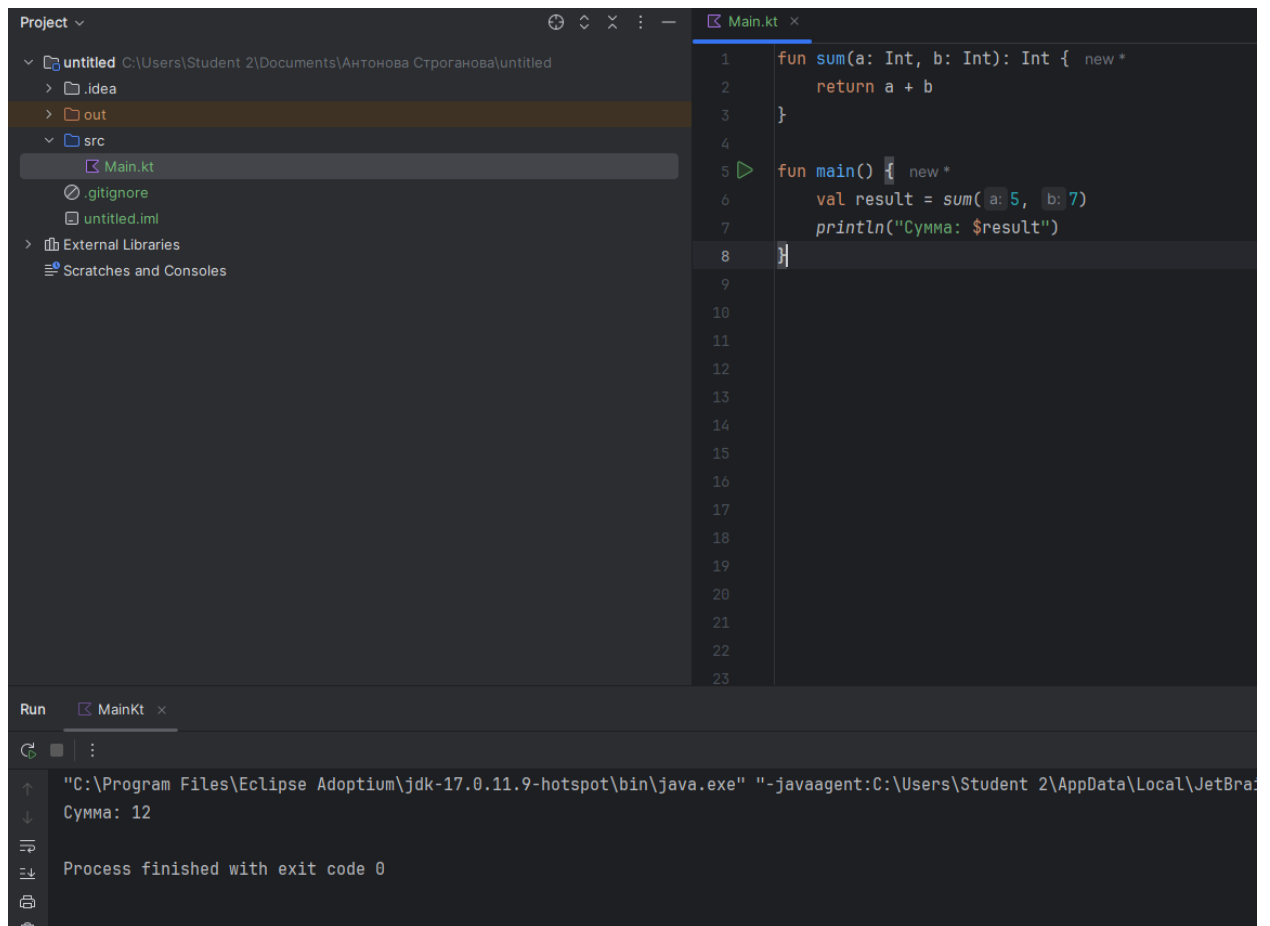
```
}
```



14.

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

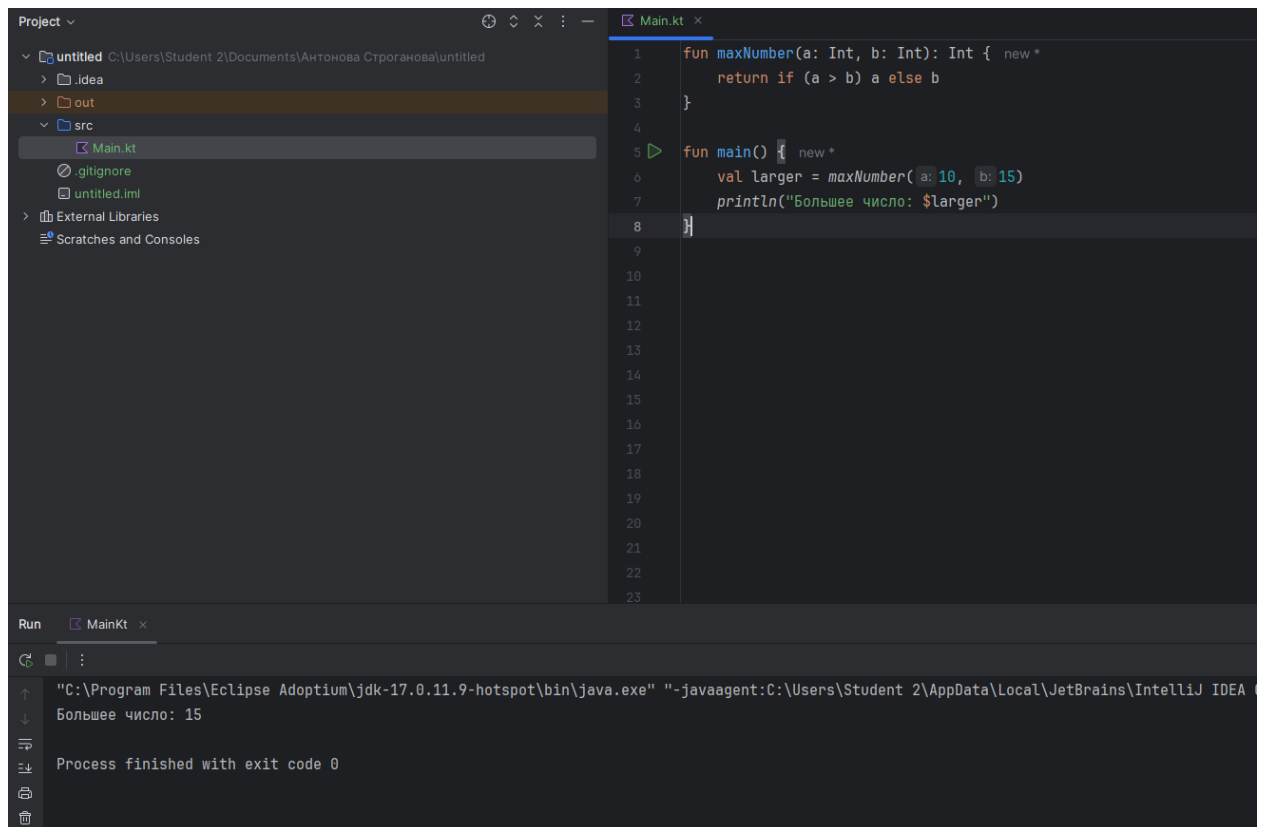
```
fun main() {  
    val result = sum(5, 7)  
    println("Сумма: $result")  
}
```



15.

```
fun maxNumber(a: Int, b: Int): Int {  
    return if (a > b) a else b  
}
```

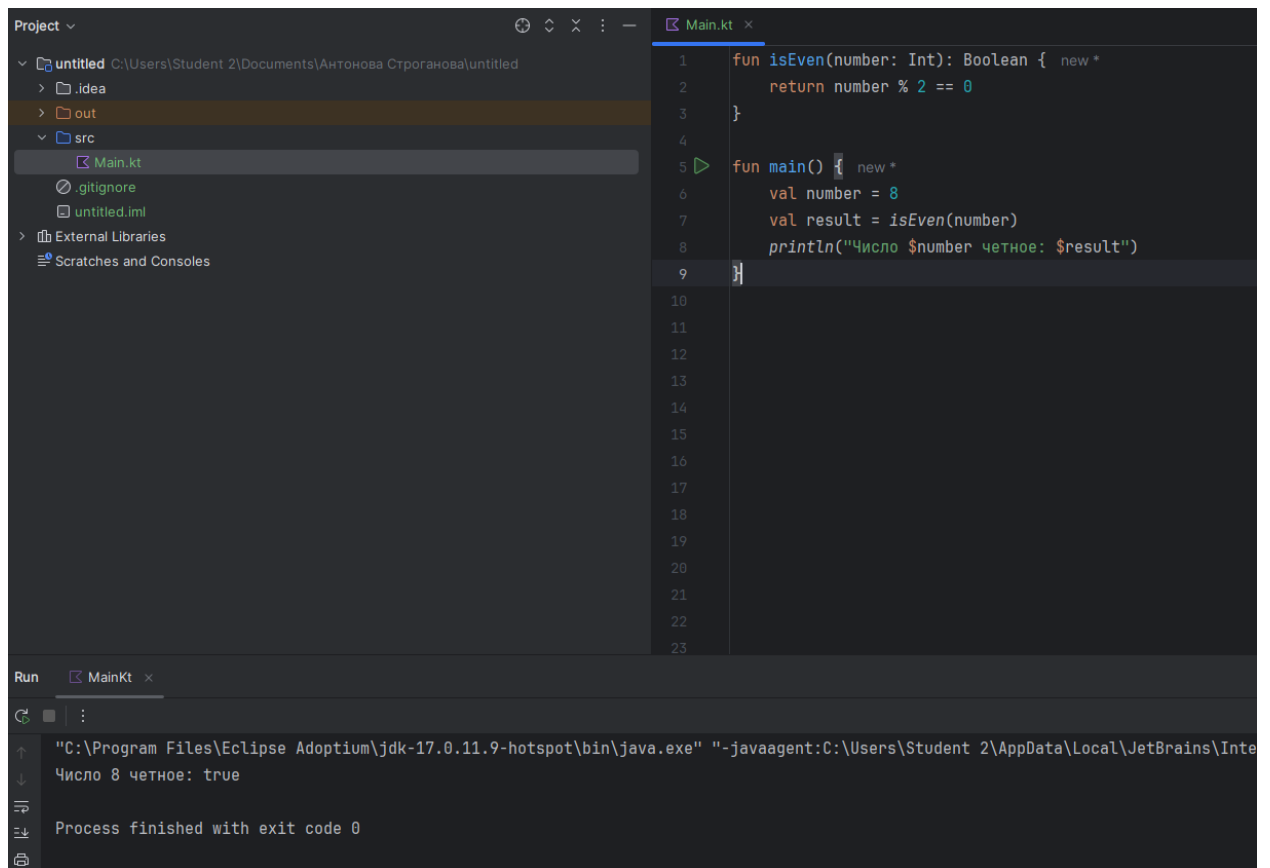
```
fun main() {  
    val larger = maxNumber(10, 15)  
    println("Большее число: $larger")  
}
```



16.

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

```
fun main() {
    val number = 8
    val result = isEven(number)
    println("Число $number четное: $result")
}
```



17.

```
fun factorial(n: Int): Long {
```

```
    if (n < 0) throw IllegalArgumentException("Факториал отрицательного числа не  
определен")
```

```
    return if (n == 0) 1 else n * factorial(n - 1)
```

```
}
```

```
fun main() {
```

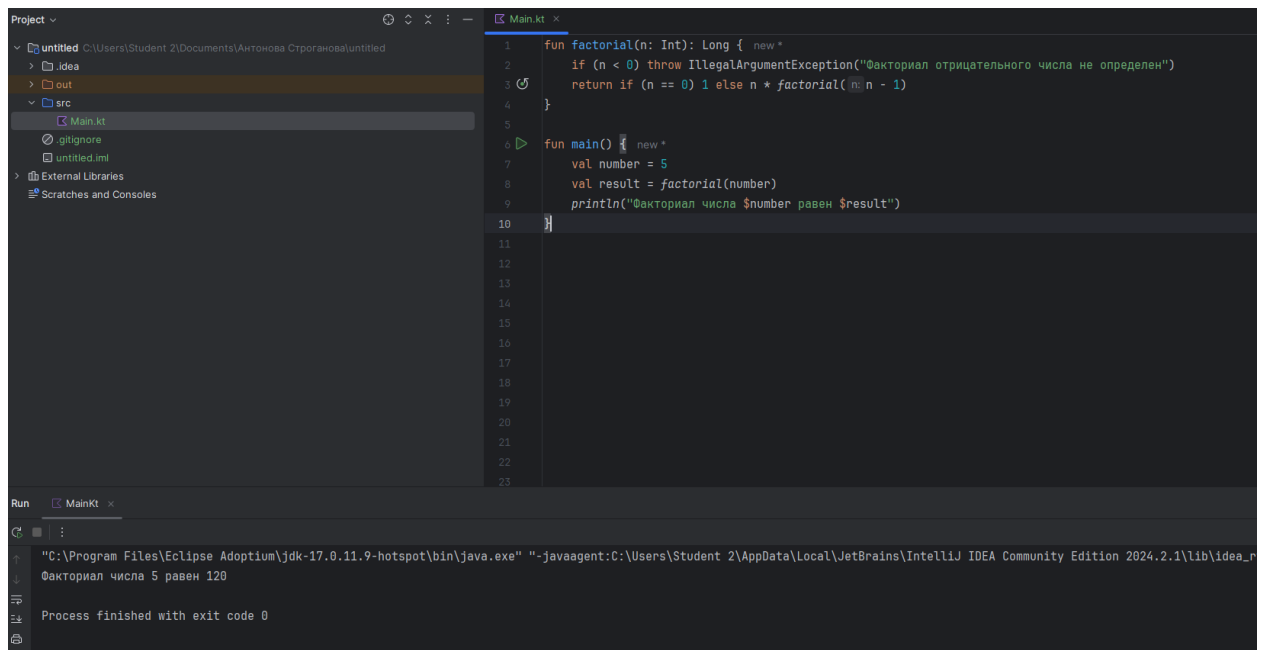
```
    val number = 5
```

```
    val result = factorial(number)
```

```
    println("Факториал числа $number равен $result")
```

```
}
```

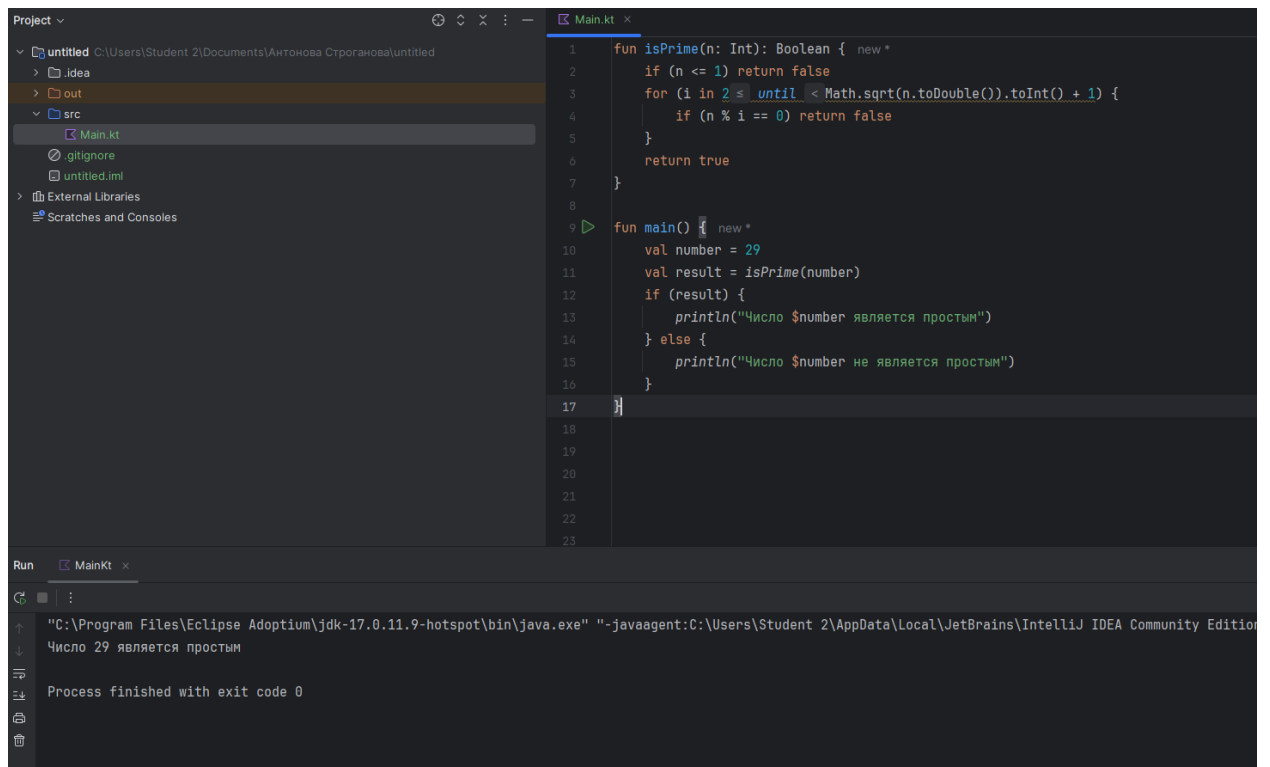




18.

```
fun isPrime(n: Int): Boolean {
    if (n <= 1) return false
    for (i in 2 until Math.sqrt(n.toDouble()).toInt() + 1) {
        if (n % i == 0) return false
    }
    return true
}
```

```
fun main() {
    val number = 29
    val result = isPrime(number)
    if (result) {
        println("Число $number является простым")
    } else {
        println("Число $number не является простым")
    }
}
```



19.

```
fun sumOfArray(array: IntArray): Int {
```

```
    var sum = 0
```

```
    for (number in array) {
```

```
        sum += number
```

```
    }
```

```
    return sum
```

```
}
```

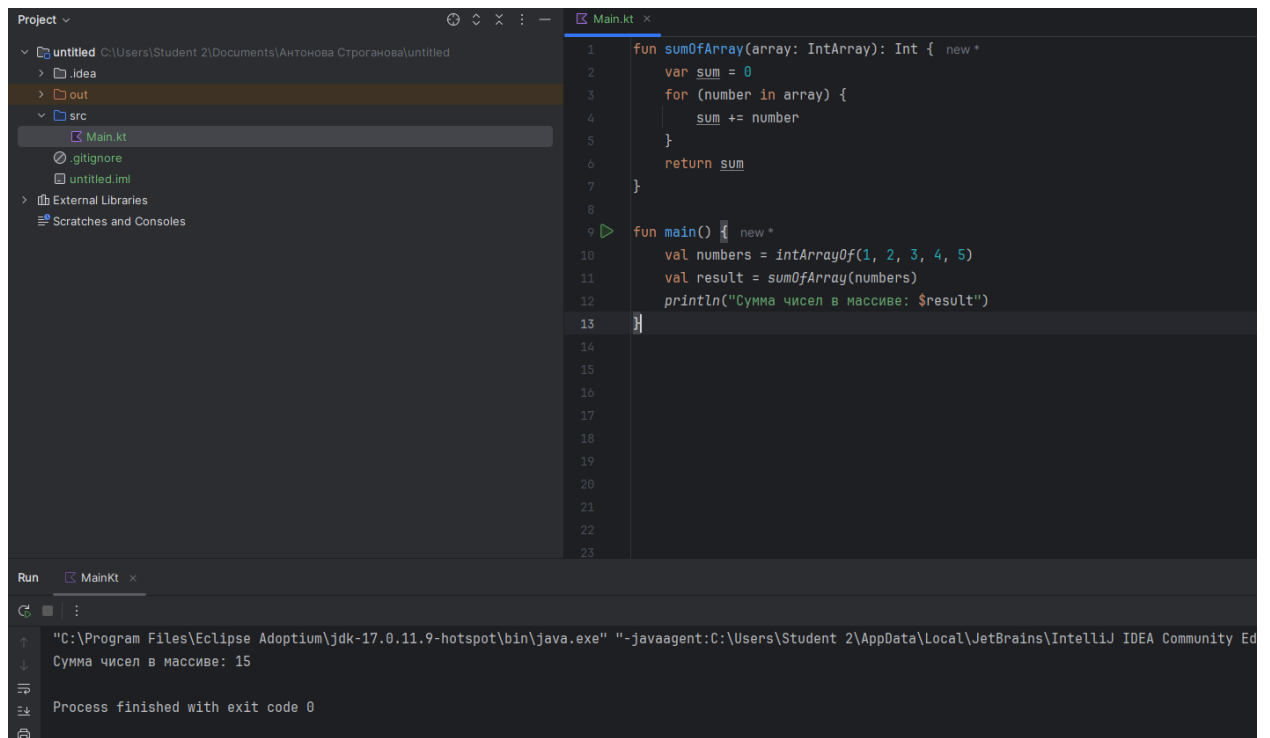
```
fun main() {
```

```
    val numbers = intArrayOf(1, 2, 3, 4, 5)
```

```
    val result = sumOfArray(numbers)
```

```
    println("Сумма чисел в массиве: $result")
```

```
}
```



20.

```
fun maxInArray(array: IntArray): Int {
```

```
    var max = array[0]
```

```
    for (number in array) {
```

```
        if (number > max) {
```

```
            max = number
```

```
        }
```

```
    }
```

```
    return max
```

```
}
```

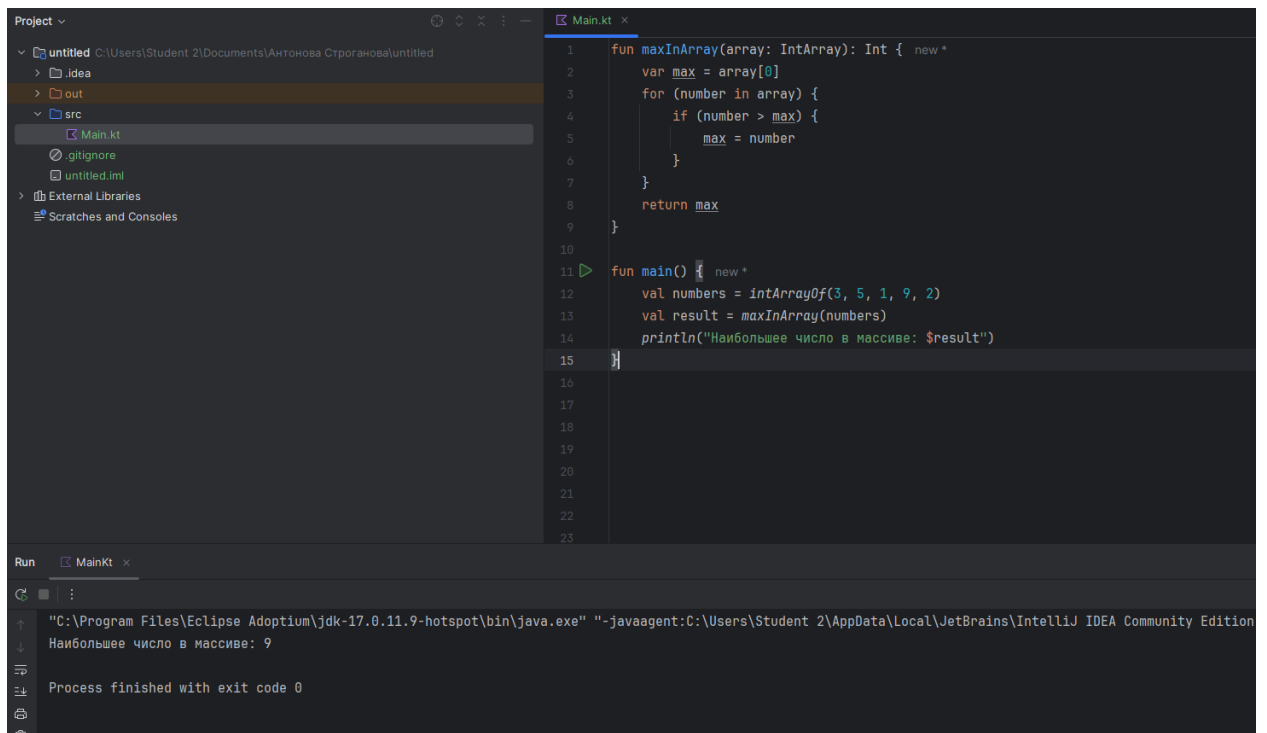
```
fun main() {
```

```
    val numbers = intArrayOf(3, 5, 1, 9, 2)
```

```
    val result = maxInArray(numbers)
```

```
    println("Наибольшее число в массиве: $result")
```

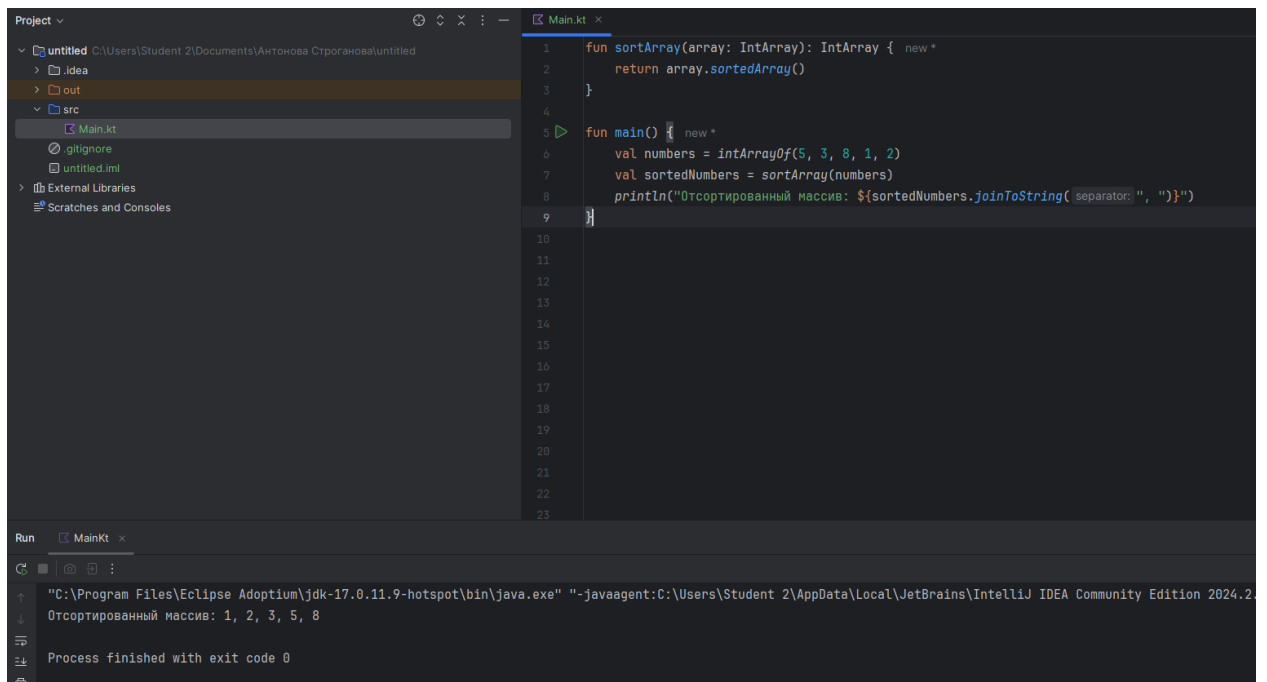
```
}
```



21.

```
fun sortArray(array: IntArray): IntArray {  
    return array.sortedArray()  
}
```

```
fun main() {  
    val numbers = intArrayOf(5, 3, 8, 1, 2)  
    val sortedNumbers = sortArray(numbers)  
    println("Отсортированный массив: ${sortedNumbers.joinToString(", ")}")  
}
```



22.

```
fun isPalindrome(text: String): Boolean {
```

```
    val cleanedText = text.replace("""\W+""".toRegex(), "").toLowerCase()
```

```
    return cleanedText == cleanedText.reversed()
```

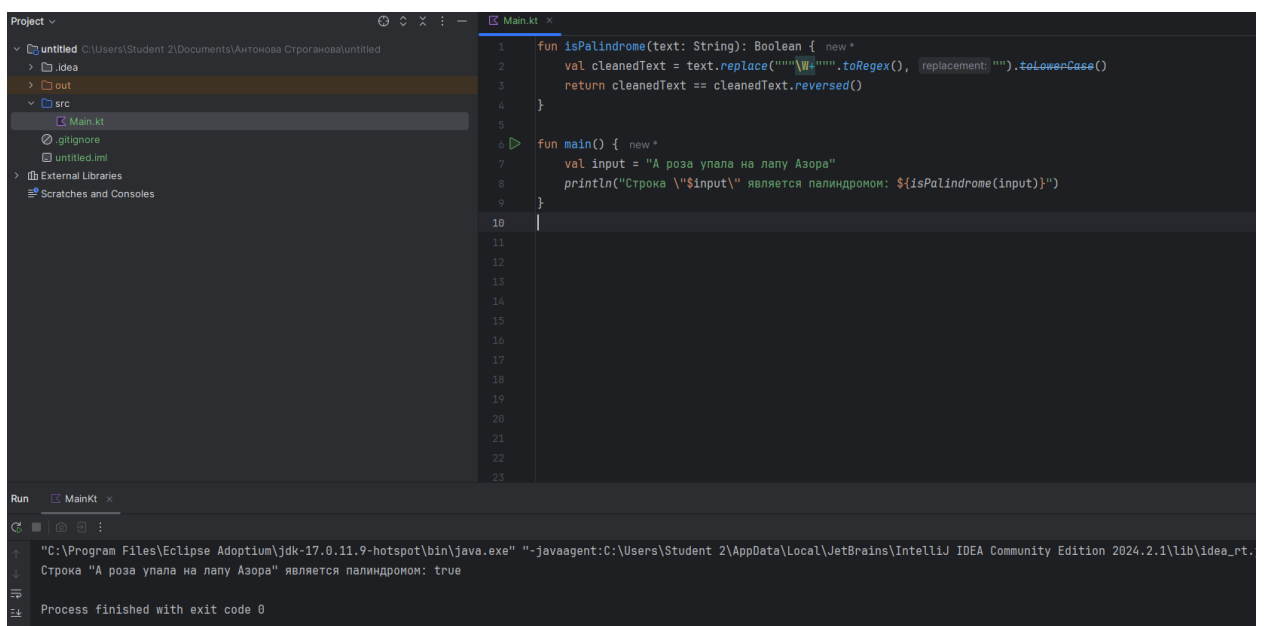
```
}
```

```
fun main() {
```

```
    val input = "А роза упала на лапу Азора"
```

```
    println("Строка \"\$input\" является палиндромом: ${isPalindrome(input)}")
```

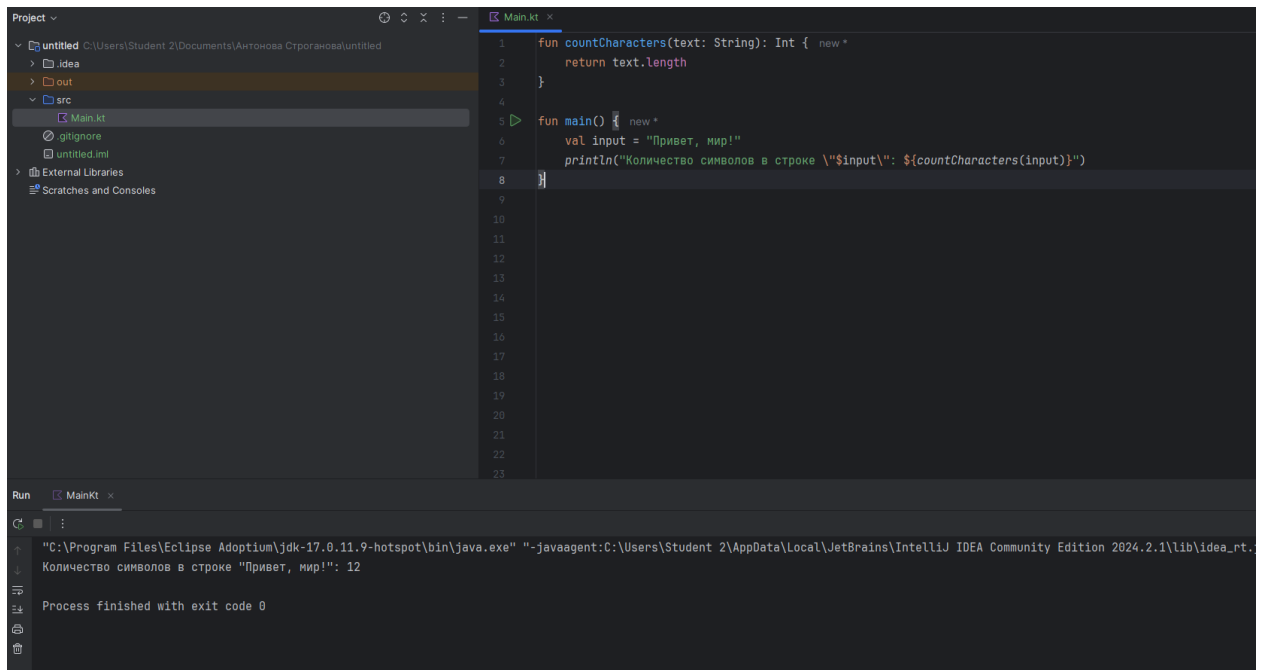
```
}
```



23.

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

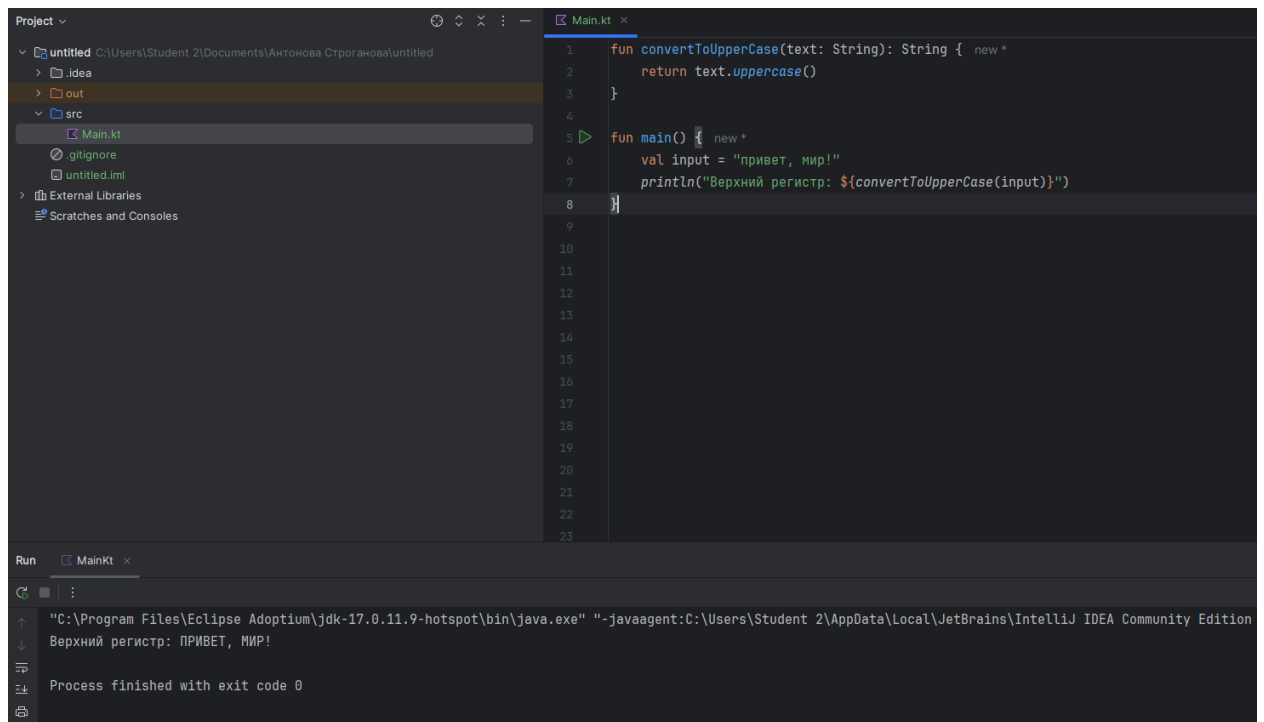
```
fun main() {  
    val input = "Привет, мир!"  
    println("Количество символов в строке \"$input\": ${countCharacters(input)}")  
}
```



24.

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

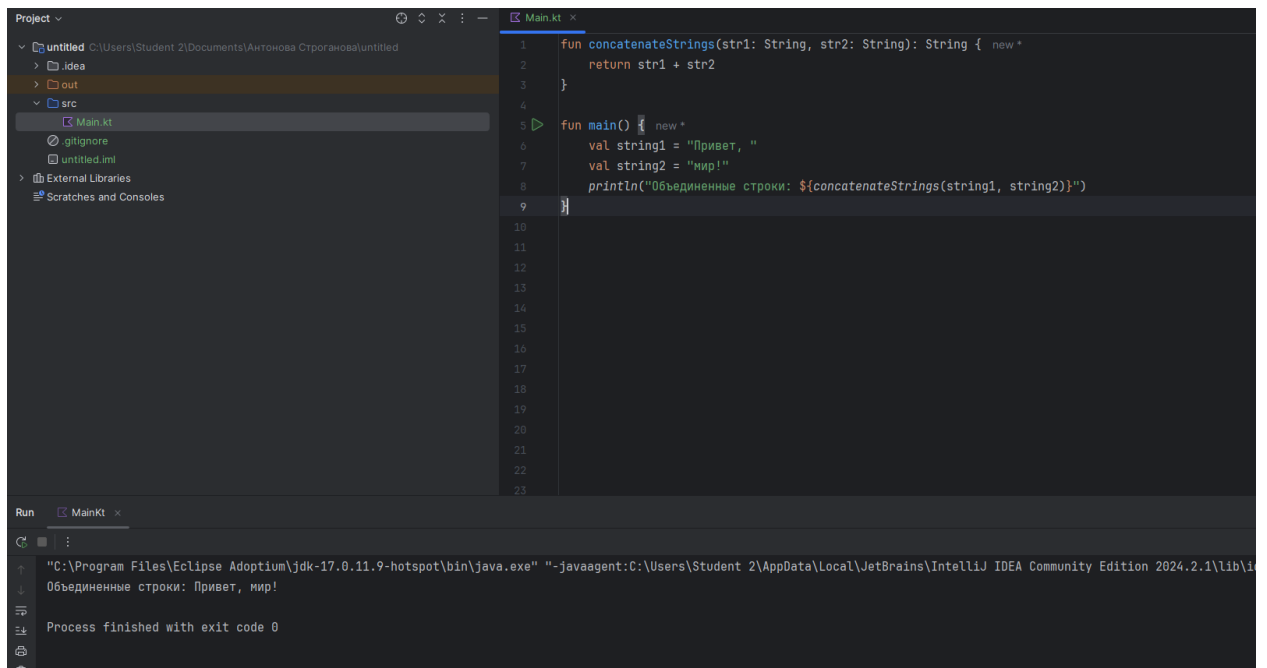
```
fun main() {  
    val input = "привет, мир!"  
    println("Верхний регистр: ${convertToUpperCase(input)}")  
}
```



25.

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

```
fun main() {  
    val string1 = "Привет, "  
    val string2 = "мир!"  
    println("Объединенные строки: ${concatenateStrings(string1, string2)}")  
}
```

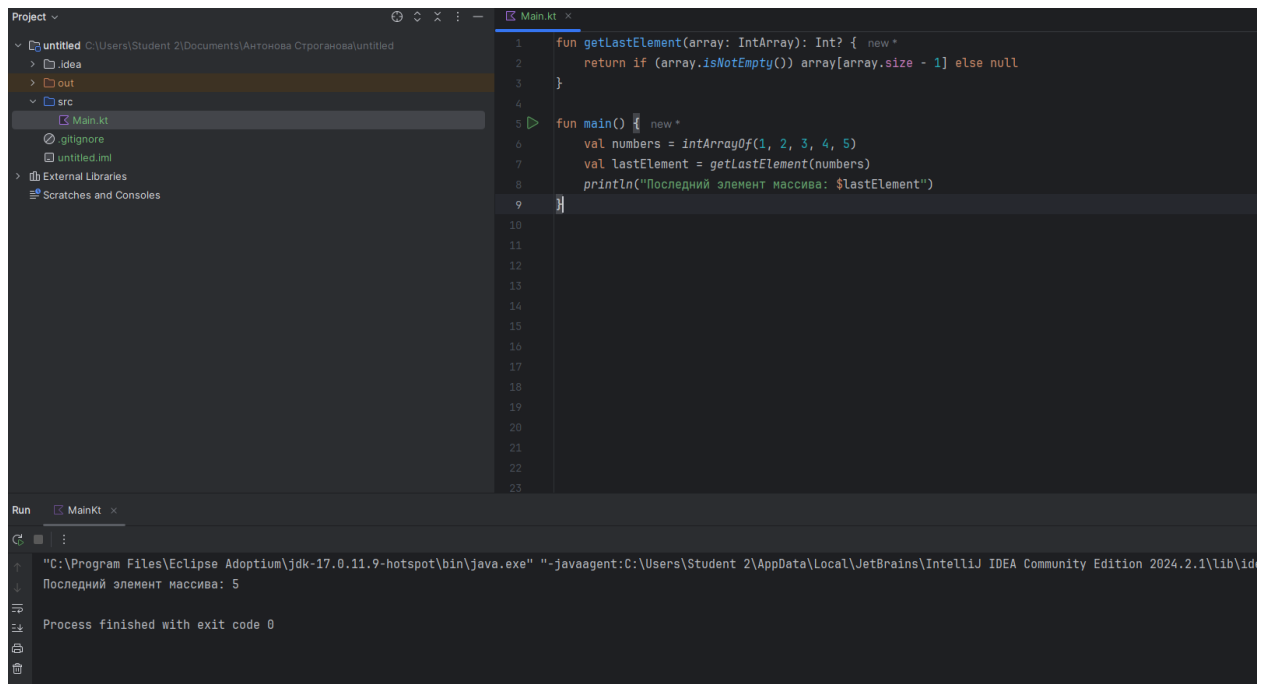


26.

```
fun getLastElement(array: IntArray): Int? {  
    return if (array.isNotEmpty()) array[array.size - 1] else null  
}
```

```
fun main() {  
    val numbers = intArrayOf(1, 2, 3, 4, 5)  
    val lastElement = getLastElement(numbers)  
    println("Последний элемент массива: $lastElement")  
}
```

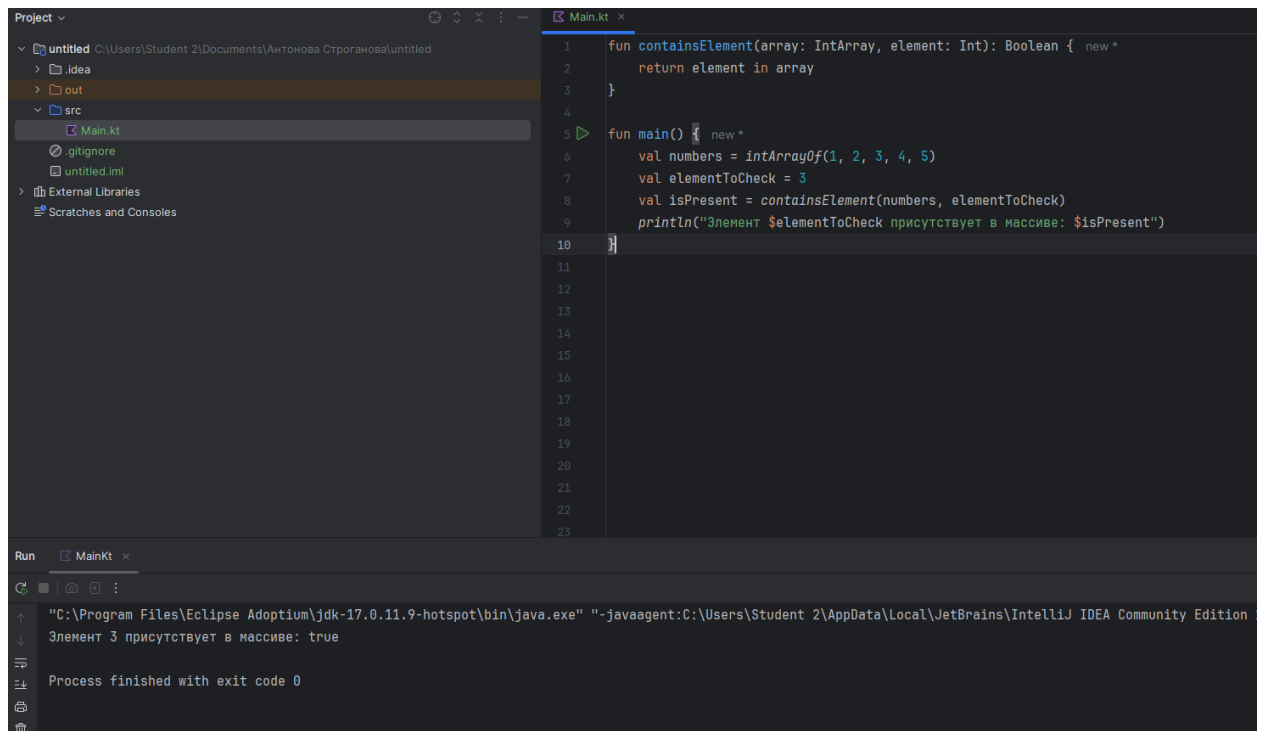




27.

```
fun containsElement(array: IntArray, element: Int): Boolean {  
    return element in array  
}
```

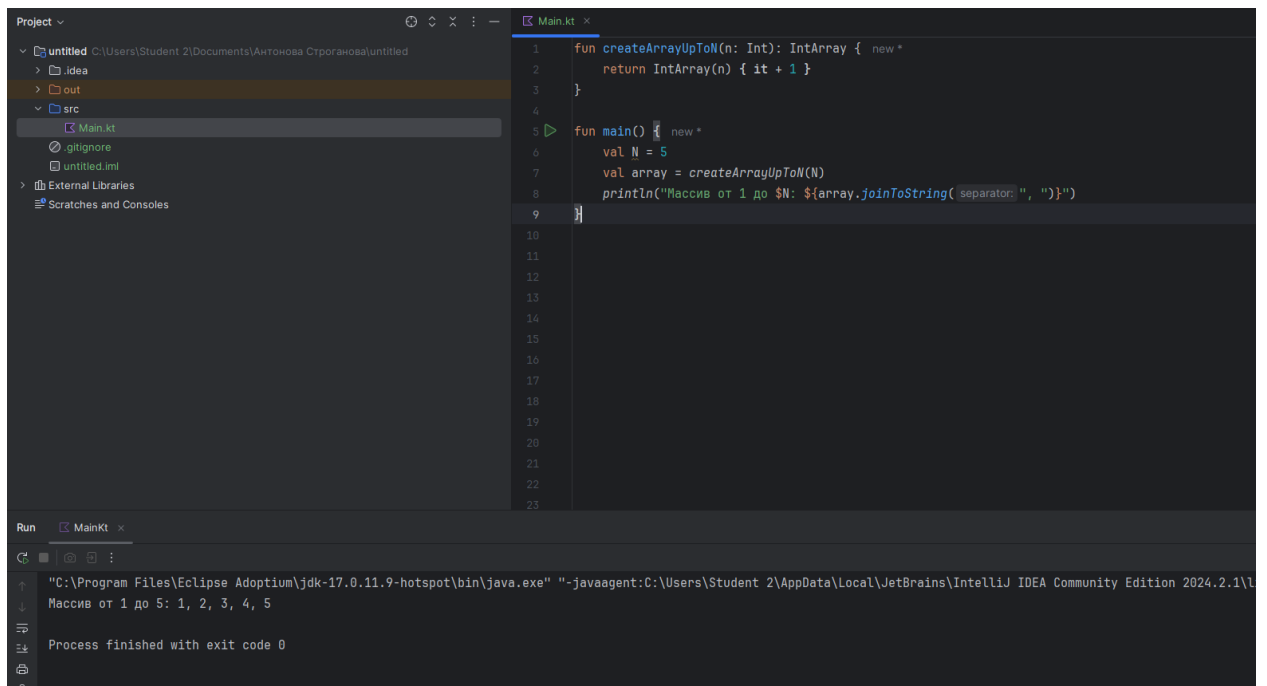
```
fun main() {  
    val numbers = intArrayOf(1, 2, 3, 4, 5)  
    val elementToCheck = 3  
    val isPresent = containsElement(numbers, elementToCheck)  
    println("Элемент $elementToCheck присутствует в массиве: $isPresent")  
}
```



28.

```
fun createArrayUpToN(n: Int): IntArray {  
    return IntArray(n) { it + 1 }  
}
```

```
fun main() {  
    val N = 5  
    val array = createArrayUpToN(N)  
    println("Массив от 1 до $N: ${array.joinToString(", ")}")  
}
```



29.

```
fun findMaxMin(arr: IntArray): Pair<Int, Int>? {
```

```
    if (arr.isEmpty()) return null
```

```
    var max = arr[0]
```

```
    var min = arr[0]
```

```
    for (num in arr) {
```

```
        if (num > max) max = num
```

```
        if (num < min) min = num
```

```
    }
```

```
    return Pair(max, min)
```

```
}
```

```
fun main() {
```

```
    val array = intArrayOf(3, 1, 4, 1, 5, 9, 2)
```

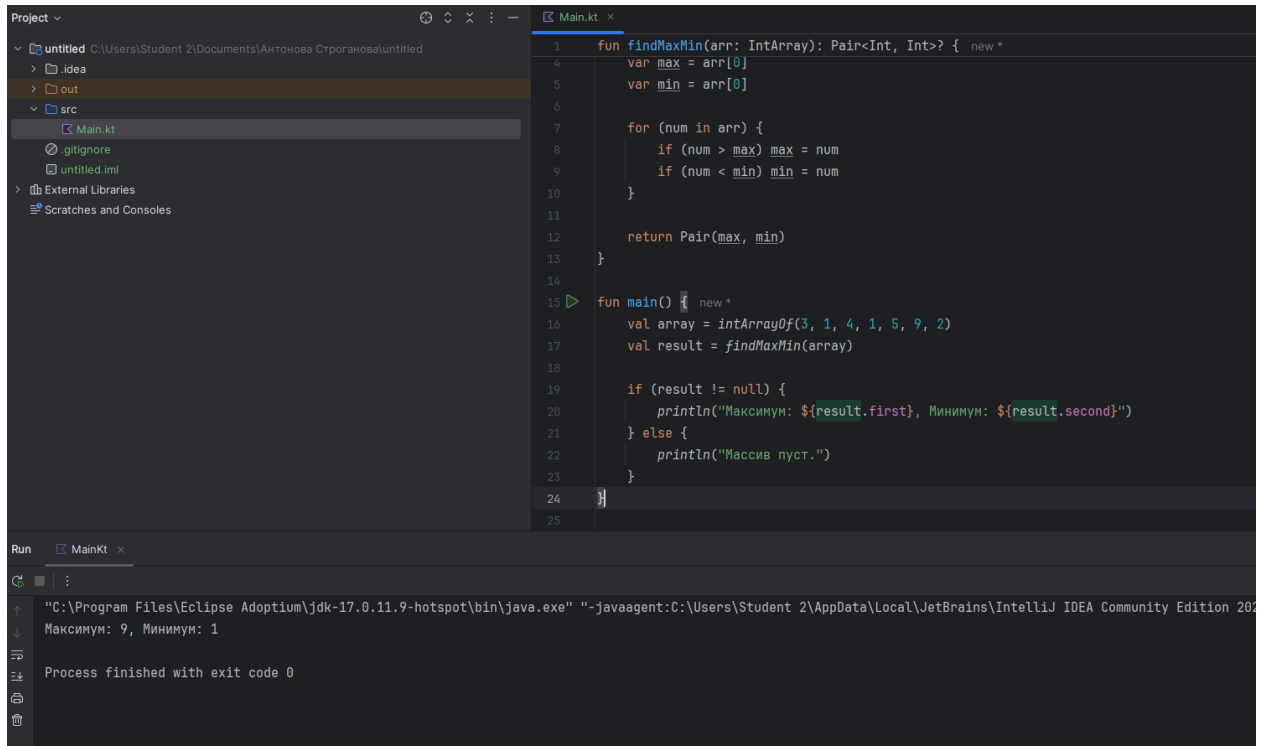
```
    val result = findMaxMin(array)
```

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

```

        println("Максимум: ${result.first}, Минимум: ${result.second}")
    } else {
        println("Массив пуст.")
    }
}

```



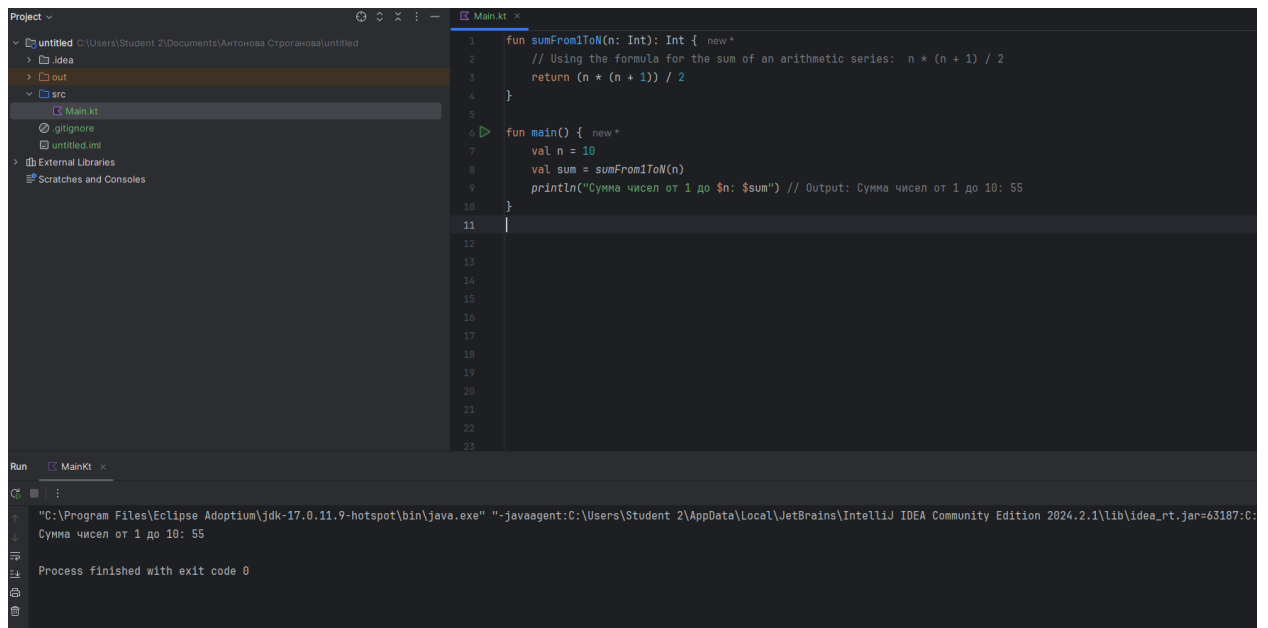
30.

```

fun sumFrom1ToN(n: Int): Int {
    // Using the formula for the sum of an arithmetic series: n * (n + 1) / 2
    return (n * (n + 1)) / 2
}

fun main() {
    val n = 10
    val sum = sumFrom1ToN(n)
    println("Сумма чисел от 1 до $n: $sum") // Output: Сумма чисел от 1 до 10: 55
}

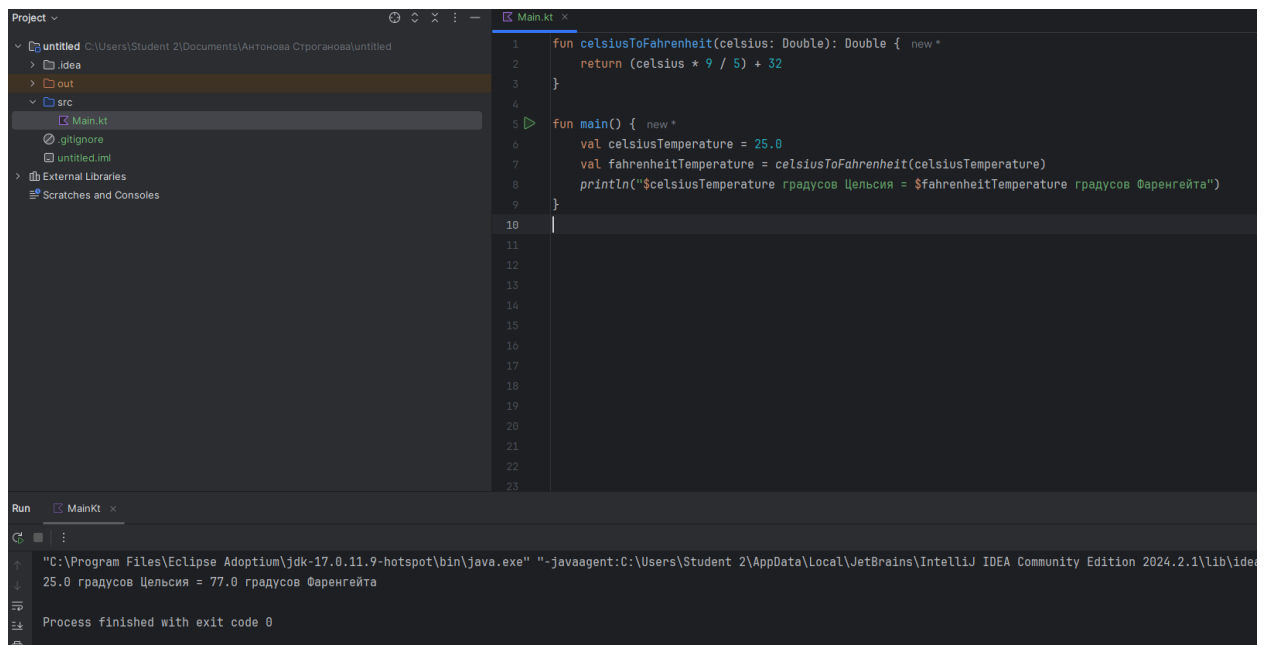
```



31.

```
fun celsiusToFahrenheit(celsius: Double): Double {  
    return (celsius * 9 / 5) + 32  
}
```

```
fun main() {  
    val celsiusTemperature = 25.0  
    val fahrenheitTemperature = celsiusToFahrenheit(celsiusTemperature)  
    println("$celsiusTemperature градусов Цельсия = $fahrenheitTemperature градусов  
Фаренгейта")  
}
```



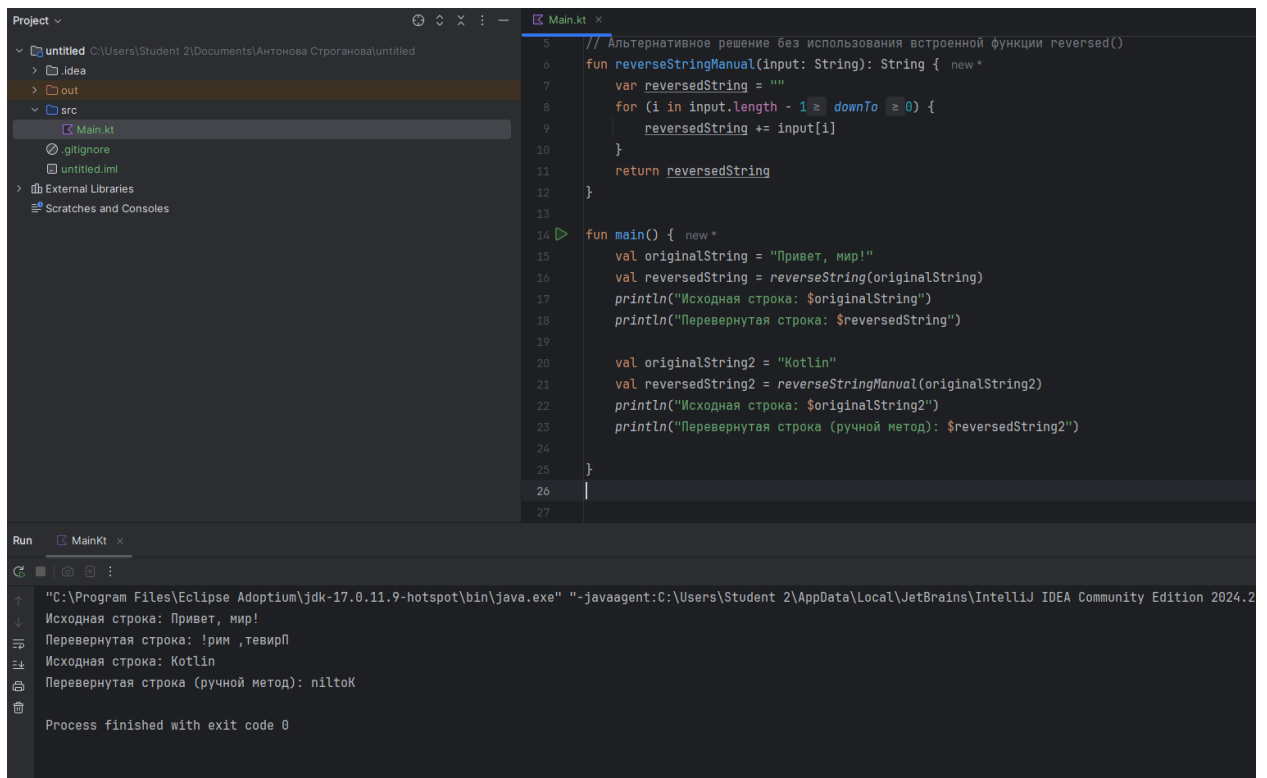
32.

```
fun reverseString(input: String): String {  
    return input.reversed()  
}
```

// Альтернативное решение без использования встроенной функции reversed()

```
fun reverseStringManual(input: String): String {  
    var reversedString = ""  
    for (i in input.length - 1 downTo 0) {  
        reversedString += input[i]  
    }  
    return reversedString  
}
```

```
fun main() {  
    val originalString = "Привет, мир!"  
    val reversedString = reverseString(originalString)  
    println("Исходная строка: $originalString")  
    println("Перевернутая строка: $reversedString")  
  
    val originalString2 = "Kotlin"  
    val reversedString2 = reverseStringManual(originalString2)  
    println("Исходная строка: $originalString2")  
    println("Перевернутая строка (ручной метод): $reversedString2")  
}
```



33.

```
fun получитьЭлементПоИндексу(массив: Array<Int>, индекс: Int): Int? {
```

```
    if (индекс >= 0 && индекс < массив.size) {
```

```
        return массив[индекс]
```

```
    } else {
```

```
        return null // Индекс за пределами массива
```

```
    }
```

```
}
```

```
fun main() {
```

```
    val числа = arrayOf(10, 20, 30, 40, 50)
```

```
    val элемент = получитьЭлементПоИндексу(числа, 2)
```

```
    if (элемент != null) {
```

```
        println("Элемент по индексу 2: $элемент") // Вывод: Элемент по индексу 2: 30
```

```
    } else {
```

```
        println("Индекс находится за пределами массива")
```

```
    }
```

```

    val элементЗаПределами = получитьЭлементПоИндексу(числа, 5) // Индекс за
пределами массива

    if (элементЗаПределами != null) {

        println("Элемент по индексу 5: $элементЗаПределами")

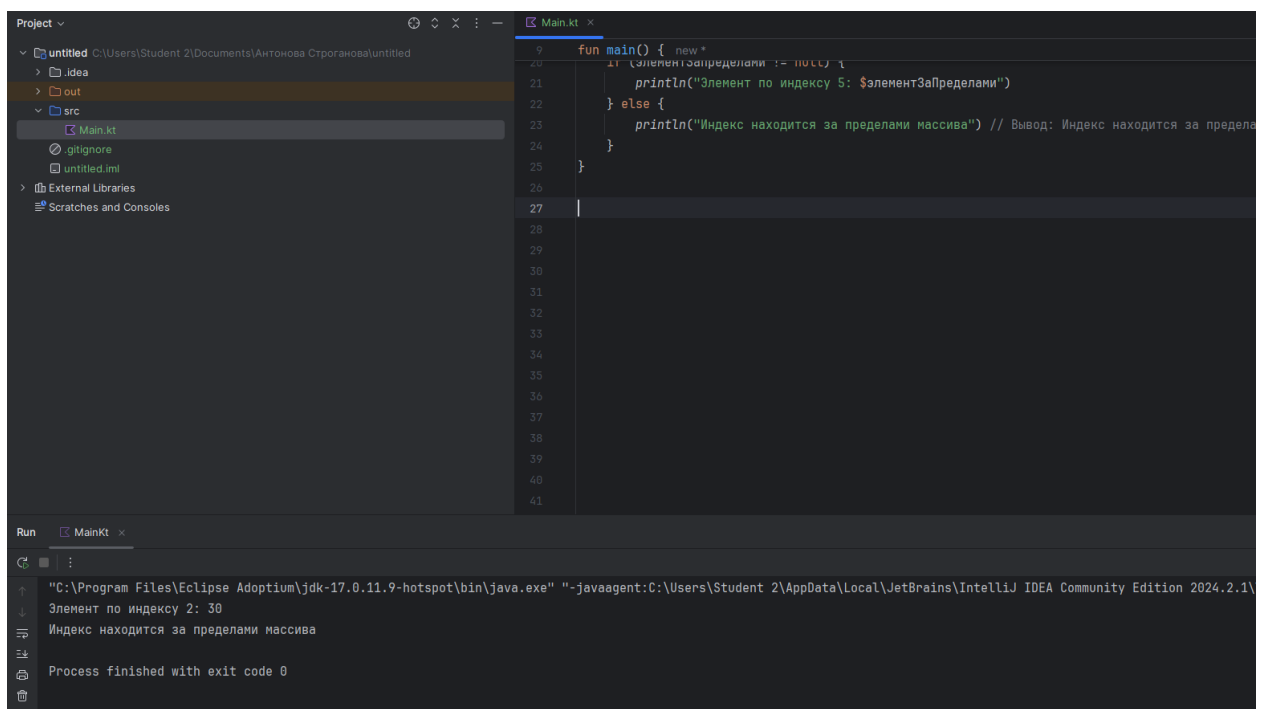
    } else {

        println("Индекс находится за пределами массива") // Вывод: Индекс находится за
пределами массива

    }

}

```



34.

```

fun removeSpaces(input: String): String {

    return input.replace(" ", "")

}

```

// Альтернативный способ с использованием регулярного выражения (более гибкий)

```

fun removeSpacesRegex(input: String): String {

    return input.replace(Regex("\\s+"), "")

}

```



```

fun main() {

    val stringWithSpaces = " Hello World! "

    val stringWithoutSpaces = removeSpaces(stringWithSpaces)

    println("Строка с пробелами: '$stringWithSpaces'")

    println("Строка без пробелов: '$stringWithoutSpaces'")


    val stringWithVariousSpaces = "Это строка\tс различными\nпробельными символами."

    val stringWithoutSpacesRegex = removeSpacesRegex(stringWithVariousSpaces)

    println("Строка с различными пробелами: '$stringWithVariousSpaces'")

    println("Строка без пробелов (Regex): '$stringWithoutSpacesRegex'")

}

```

The screenshot shows an IDE with a project named 'untitled'. The source file 'Main.kt' contains the following code:

```

1 fun removeSpaces(input: String): String { new *
2     return input.replace( oldValue: " ", newValue: "")
3 }
4
5 // Альтернативный способ с использованием регулярного выражения (более гибкий)
6 fun removeSpacesRegex(input: String): String { new *
7     return input.replace(Regex( pattern: "\\s+" ), replacement: "")
8 }
9
10 fun main() { new *
11     val stringWithSpaces = " Hello World! "
12     val stringWithoutSpaces = removeSpaces(stringWithSpaces)
13     println("Строка с пробелами: '$stringWithSpaces'")
14     println("Строка без пробелов: '$stringWithoutSpaces'")
15
16     val stringWithVariousSpaces = "Это строка\tс различными\nпробельными символами."
17     val stringWithoutSpacesRegex = removeSpacesRegex(stringWithVariousSpaces)
18     println("Строка с различными пробелами: '$stringWithVariousSpaces'")
19     println("Строка без пробелов (Regex): '$stringWithoutSpacesRegex'")
20 }
21
22
23

```

The Run console shows the following output:

```

"C:\Program Files\Eclipse Adoptium\jdk-17.0.11.9-hotspot\bin\java.exe" "-javaagent:C:\Users\Student 2\AppData\Local\JetBrains\IntelliJ IDEA Community Edition 2024.2.1\lib\idea_rt.
Строка с пробелами: ' Hello World! '
Строка без пробелов: 'HelloWorld!'
Строка с различными пробелами: 'Это строка с различными
пробельными символами.'
Строка без пробелов (Regex): 'Этострокасразличнымпробельнымисимволами.'
Process finished with exit code 0

```

35.

```

fun sumOfFirstN(n: Int): Int {

    if (n <= 0) {

        return 0 // Обрабатываем случай, когда N не является натуральным числом

    }

    return n * (n + 1) / 2

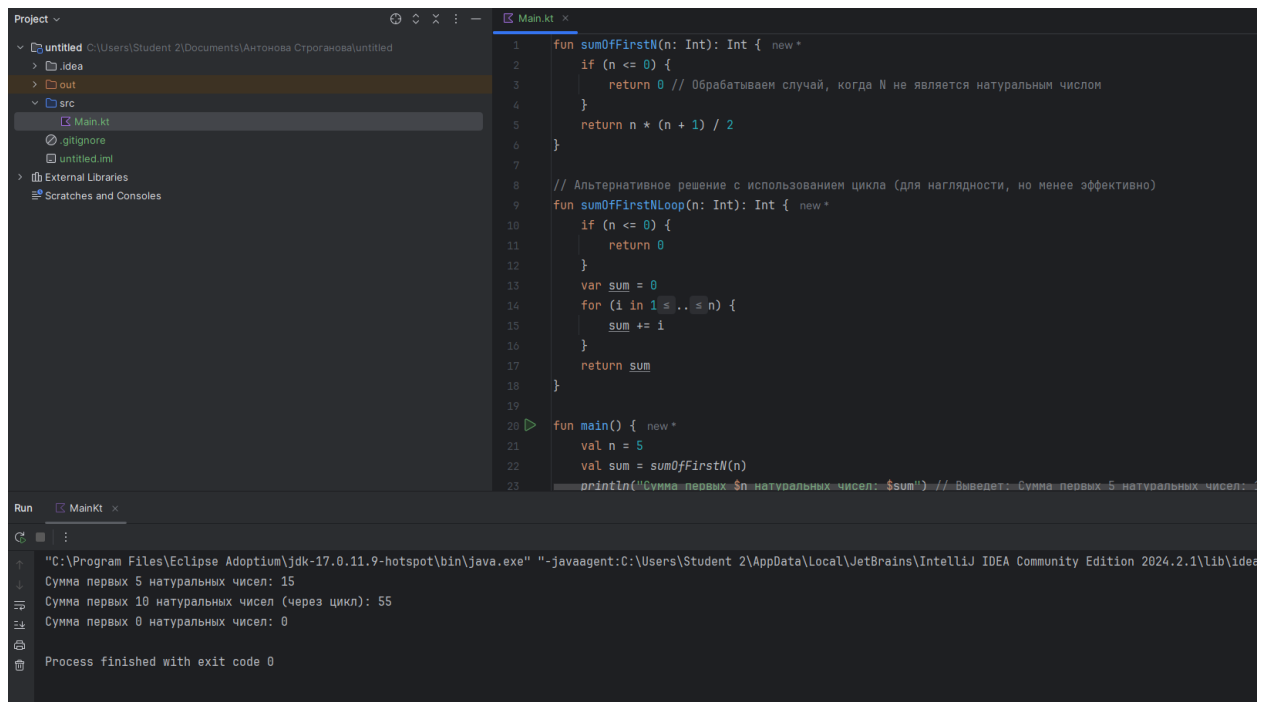
}

```

// Альтернативное решение с использованием цикла (для наглядности, но менее эффективно)

```
fun sumOfFirstNLoop(n: Int): Int {  
    if (n <= 0) {  
        return 0  
    }  
    var sum = 0  
    for (i in 1..n) {  
        sum += i  
    }  
    return sum  
}
```

```
fun main() {  
    val n = 5  
    val sum = sumOfFirstN(n)  
    println("Сумма первых $n натуральных чисел: $sum") // Выведет: Сумма первых 5  
    натуральных чисел: 15  
  
    val n2 = 10  
    val sum2 = sumOfFirstNLoop(n2)  
    println("Сумма первых $n2 натуральных чисел (через цикл): $sum2") // Выведет: Сумма  
    первых 10 натуральных чисел (через цикл): 55  
  
    val n3 = 0  
    val sum3 = sumOfFirstN(n3)  
    println("Сумма первых $n3 натуральных чисел: $sum3") // Выведет: Сумма первых 0  
    натуральных чисел: 0  
}
```



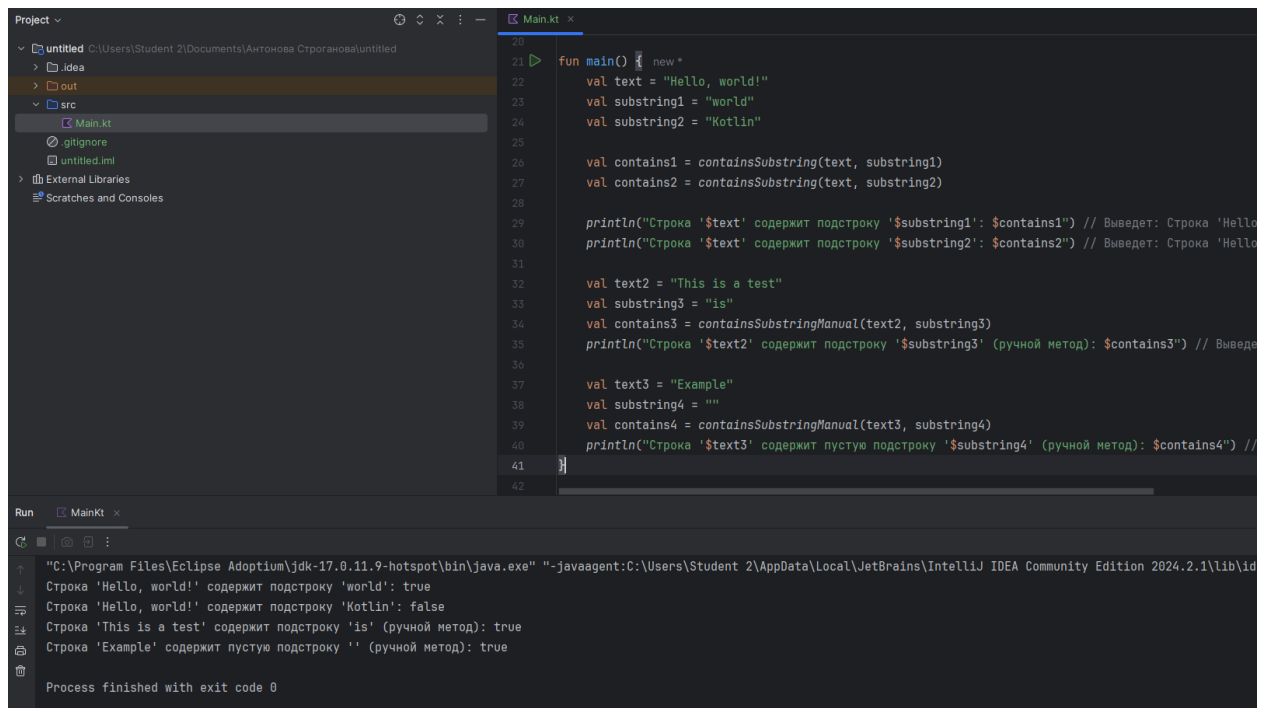
36.

```
fun containsSubstring(text: String, substring: String): Boolean {  
    return text.contains(substring)  
}
```

// Альтернативный вариант без использования встроенной функции contains()

```
fun containsSubstringManual(text: String, substring: String): Boolean {  
    if (substring.isEmpty()) {  
        return true // Пустая подстрока всегда содержится в любой строке  
    }  
  
    for (i in 0..text.length - substring.length) {  
        if (text.substring(i, i + substring.length) == substring) {  
            return true  
        }  
    }  
  
    return false  
}
```

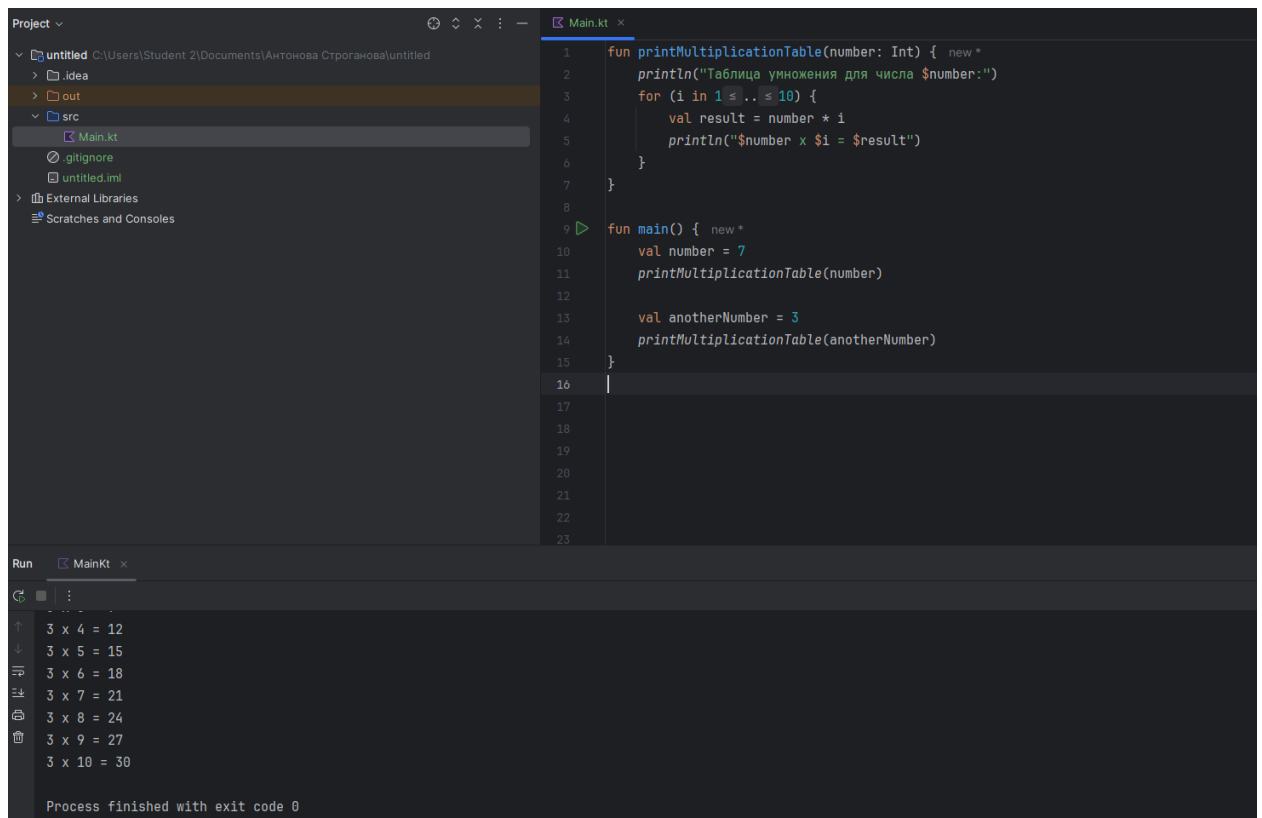
```
fun main() {  
    val text = "Hello, world!"  
    val substring1 = "world"  
    val substring2 = "Kotlin"  
  
    val contains1 = containsSubstring(text, substring1)  
    val contains2 = containsSubstring(text, substring2)  
  
    println("Строка '$text' содержит подстроку '$substring1': $contains1") // Выведет: Строка  
    'Hello, world!' содержит подстроку 'world': true  
  
    println("Строка '$text' содержит подстроку '$substring2': $contains2") // Выведет: Строка  
    'Hello, world!' содержит подстроку 'Kotlin': false  
  
    val text2 = "This is a test"  
    val substring3 = "is"  
    val contains3 = containsSubstringManual(text2, substring3)  
  
    println("Строка '$text2' содержит подстроку '$substring3' (ручной метод): $contains3") //  
    Выведет: Строка 'This is a test' содержит подстроку 'is' (ручной метод): true  
  
    val text3 = "Example"  
    val substring4 = ""  
    val contains4 = containsSubstringManual(text3, substring4)  
  
    println("Строка '$text3' содержит пустую подстроку '$substring4' (ручной метод):  
    $contains4") // Выведет: Строка 'Example' содержит пустую подстроку " (ручной метод):  
    true  
}
```



37.

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

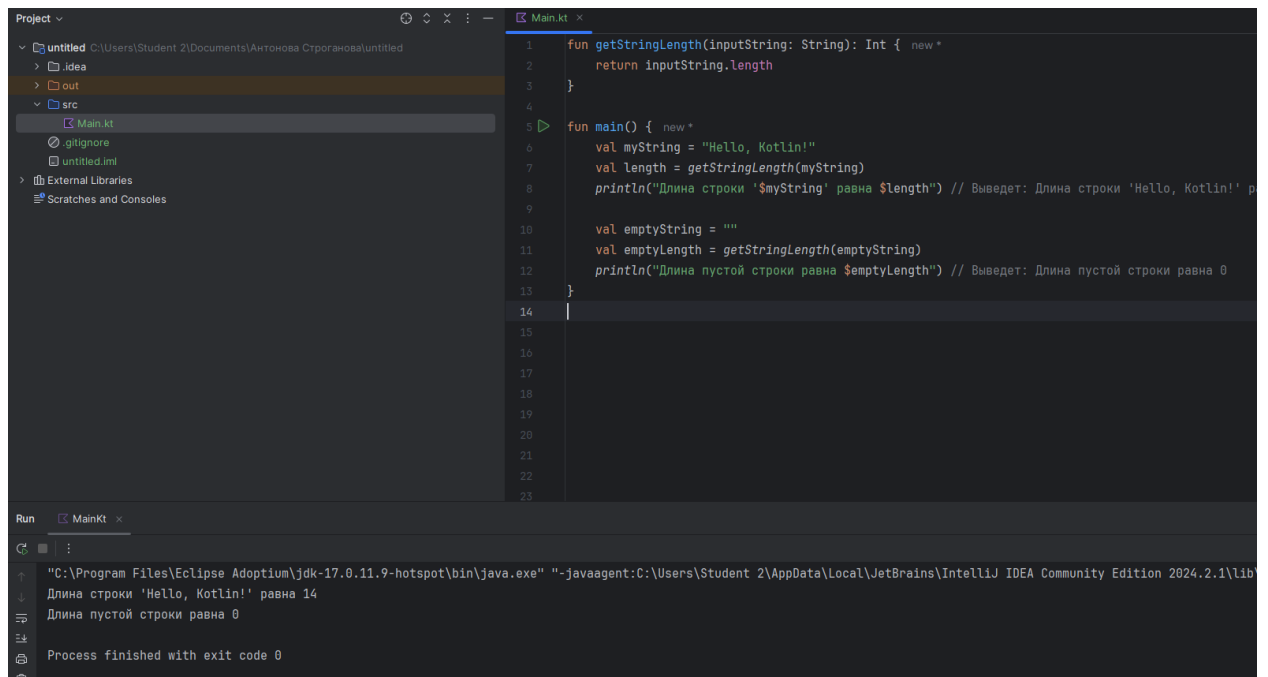
```
fun main() {  
    val number = 7  
    printMultiplicationTable(number)  
  
    val anotherNumber = 3  
    printMultiplicationTable(anotherNumber)  
}
```



38.

```
fun getStringLength(inputString: String): Int {  
    return inputString.length  
}
```

```
fun main() {  
    val myString = "Hello, Kotlin!"  
    val length = getStringLength(myString)  
    println("Длина строки '$myString' равна $length") // Выведет: Длина строки 'Hello,  
    Kotlin!' равна 14  
  
    val emptyString = ""  
    val emptyLength = getStringLength(emptyString)  
    println("Длина пустой строки равна $emptyLength") // Выведет: Длина пустой строки  
    равна 0  
}
```



39.

```
fun перевернутьМассив(массив: Array<Int>): Array<Int> {
```

```
    val длина = массив.size
```

```
    for (i in 0 until длина / 2) {
```

```
        val временнаяПеременная = массив[i]
```

```
        массив[i] = массив[длина - 1 - i]
```

```
        массив[длина - 1 - i] = временнаяПеременная
```

```
    }
```

```
    return массив
```

```
}
```

```
fun main() {
```

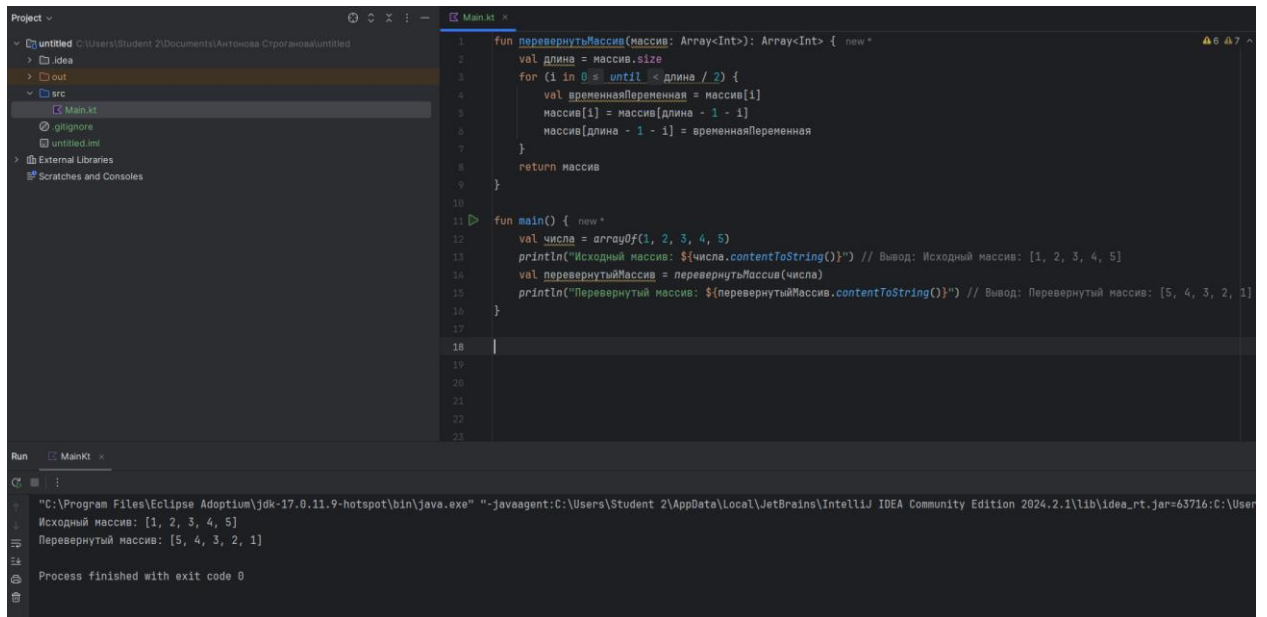
```
    val числа = arrayOf(1, 2, 3, 4, 5)
```

```
    println("Исходный массив: ${числа.contentToString()}") // Вывод: Исходный массив: [1, 2, 3, 4, 5]
```

```
    val перевернутыйМассив = перевернутьМассив(числа)
```

```
    println("Перевернутый массив: ${перевернутыйМассив.contentToString()}") // Вывод:
    Перевернутый массив: [5, 4, 3, 2, 1]
```

```
}
```



40.

```
fun скопироватьМассив(массив: Array<Int>): Array<Int> {
```

```
    val новыйМассив = массив.copyOfOf()
```

```
    return новыйМассив
```

```
}
```

```
fun main() {
```

```
    val исходныйМассив = arrayOf(1, 2, 3, 4, 5)
```

```
    val скопированныйМассив = скопироватьМассив(исходныйМассив)
```

```
    println("Исходный массив: ${исходныйМассив.contentToString()}")
```

```
    println("Скопированный массив: ${скопированныйМассив.contentToString()}")
```

```
    // Проверка, что массивы действительно разные объекты
```

```
    скопированныйМассив[0] = 100
```

```
    println("Исходный массив после изменения копии:
```

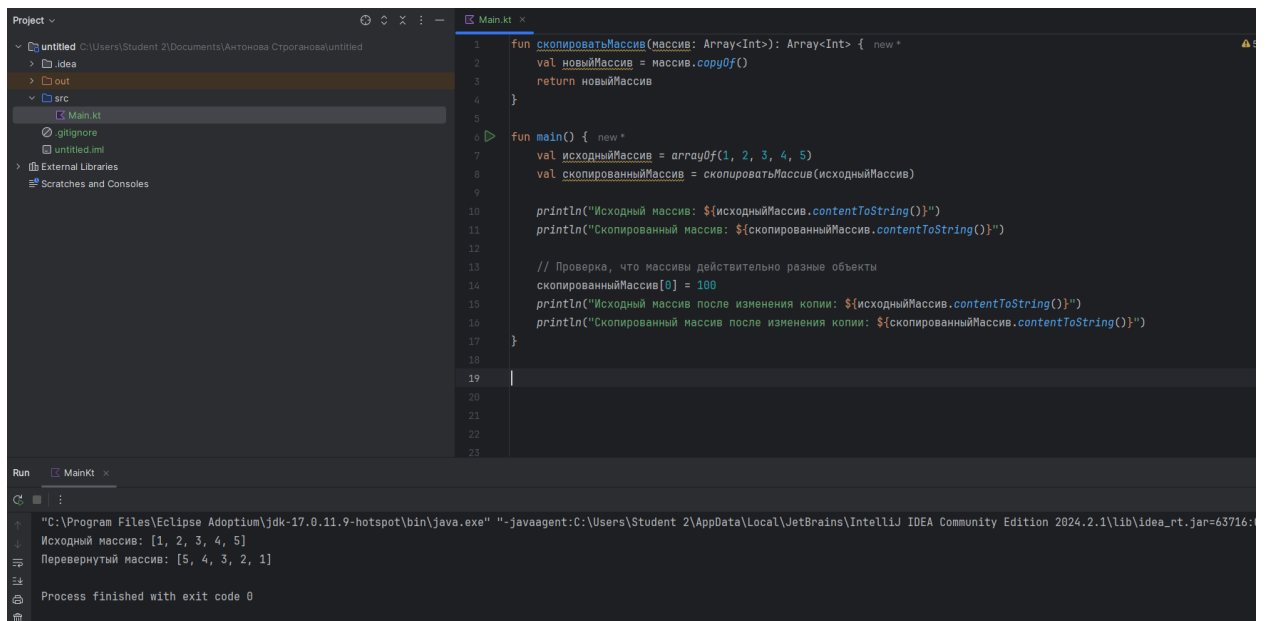
```
    ${исходныйМассив.contentToString()}")
```

```
    println("Скопированный массив после изменения копии:
```

```
    ${скопированныйМассив.contentToString()}")
```

```
}
```





41.

```
fun countVowels(inputString: String): Int {
```

```
    val vowels = "aeiouAEIOU"
```

```
    var count = 0
```

```
    for (char in inputString) {
```

```
        if (vowels.contains(char)) {
```

```
            count++
```

```
        }
```

```
    }
```

```
    return count
```

```
}
```

// Альтернативный вариант с использованием регулярного выражения

```
fun countVowelsRegex(inputString: String): Int {
```

```
    val vowelRegex = "[aeiouAEIOU]".toRegex()
```

```
    return vowelRegex.findAll(inputString).count()
```

```
}
```

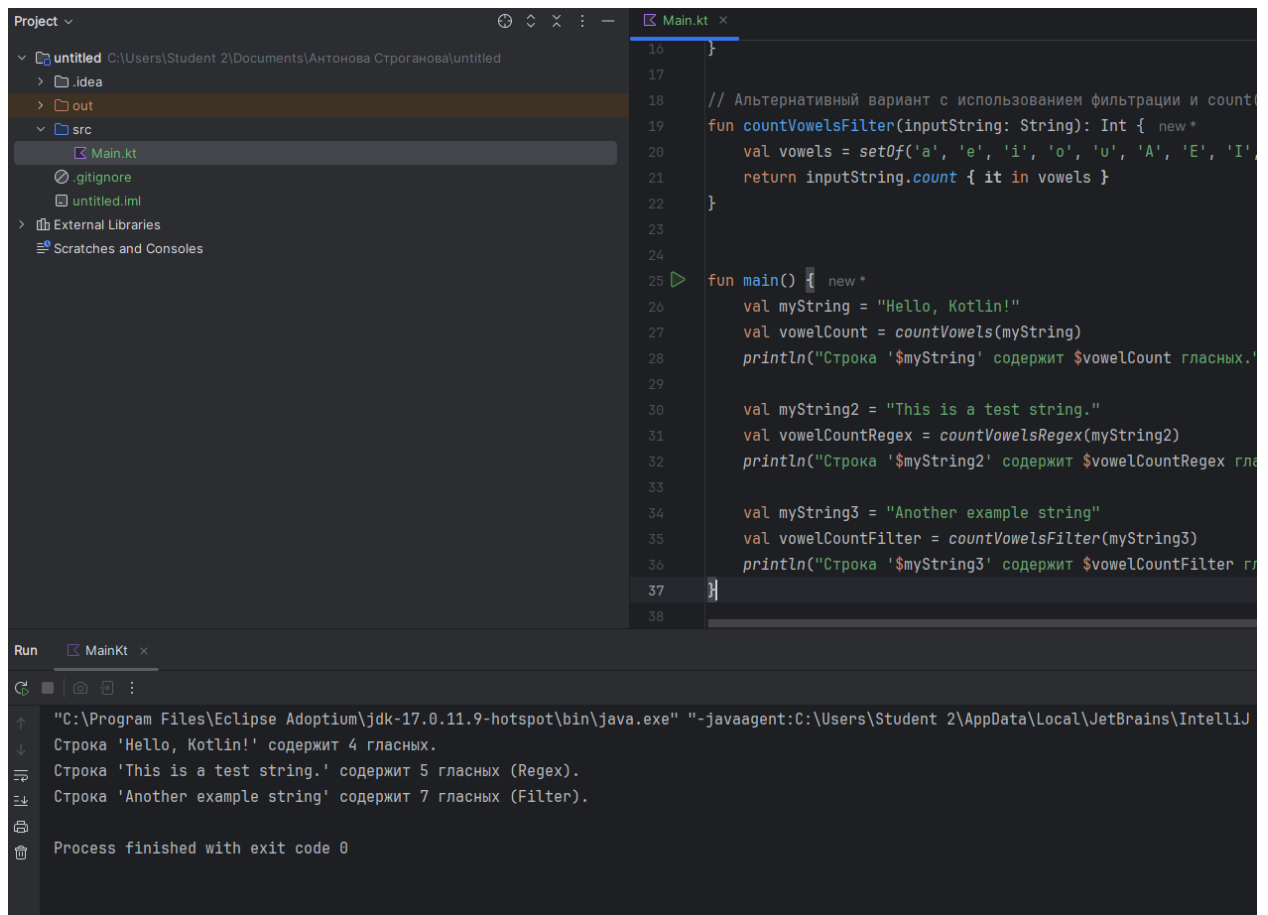
// Альтернативный вариант с использованием фильтрации и count()

```
fun countVowelsFilter(inputString: String): Int {
```

```
    val vowels = setOf('a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U')
```

```
        return inputString.count { it in vowels }  
    }  
}
```

```
fun main() {  
    val myString = "Hello, Kotlin!"  
    val vowelCount = countVowels(myString)  
    println("Строка '$myString' содержит $vowelCount гласных.") // Выведет: Строка 'Hello,  
    Kotlin!' содержит 3 гласных.  
  
    val myString2 = "This is a test string."  
    val vowelCountRegex = countVowelsRegex(myString2)  
    println("Строка '$myString2' содержит $vowelCountRegex гласных (Regex).") // Выведет:  
    Строка 'This is a test string.' содержит 4 гласных (Regex).  
  
    val myString3 = "Another example string"  
    val vowelCountFilter = countVowelsFilter(myString3)  
    println("Строка '$myString3' содержит $vowelCountFilter гласных (Filter).") // Выведет:  
    Строка 'Another example string' содержит 6 гласных (Filter).  
}
```



42.

```
fun findFirstIndex(array: IntArray, element: Int): Int {  
    for (i in array.indices) {  
        if (array[i] == element) {  
            return i // Возвращаем индекс, если элемент найден  
        }  
    }  
    return -1 // Возвращаем -1, если элемент не найден  
}
```

// Альтернативный вариант с использованием встроенной функции indexOf()

```
fun findFirstIndexBuiltIn(array: IntArray, element: Int): Int {  
    return array.indexOf(element)  
}
```

```

fun main() {
    val myArray = intArrayOf(10, 20, 30, 40, 20, 50)
    val elementToFind = 20

    val index = findFirstIndex(myArray, elementToFind)

    if (index != -1) {
        println("Элемент $elementToFind найден по индексу $index") // Выведет: Элемент 20
        найден по индексу 1
    } else {
        println("Элемент $elementToFind не найден в массиве")
    }

    val myArray2 = intArrayOf(1, 2, 3, 4, 5)
    val elementToFind2 = 6

    val index2 = findFirstIndexBuiltIn(myArray2, elementToFind2)

    if (index2 != -1) {
        println("Элемент $elementToFind2 найден по индексу $index2 (встроенная функция)")
    } else {
        println("Элемент $elementToFind2 не найден в массиве (встроенная функция)") //
        Выведет: Элемент 6 не найден в массиве (встроенная функция)
    }
}

```

