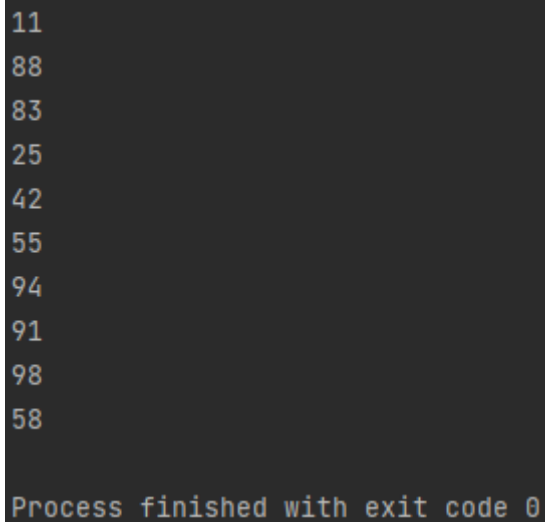


1.

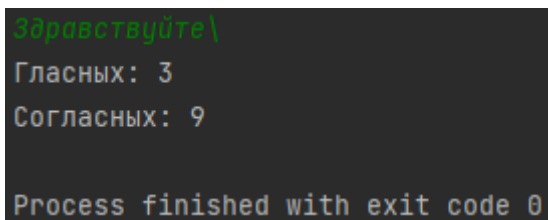
```
fun main() {  
    repeat(10) {  
        println((1..100).random())  
    }  
}
```



```
11  
88  
83  
25  
42  
55  
94  
91  
98  
58  
  
Process finished with exit code 0
```

2.

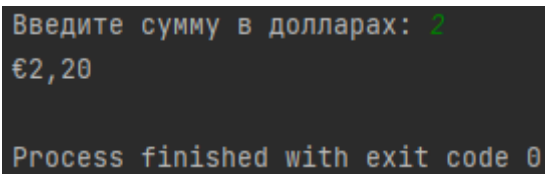
```
fun main() {  
    val input = readLine() ?: ""  
    val vowels = "аеёиоуыэюяАЕЁИОУЫЭЮЯ"  
    val vowelsCount = input.count { it in vowels }  
    val consonantsCount = input.count { it.isLetter() && it !in vowels }  
    println("Гласных: $vowelsCount")  
    println("Согласных: $consonantsCount")  
}
```



```
Здравствуйте\  
Гласных: 3  
Согласных: 9  
  
Process finished with exit code 0
```

3.

```
fun main() {  
    val rate = 1.1  
    print("Введите сумму в долларах: ")  
    val dollars = readLine()?.toDoubleOrNull() ?: 0.0  
    val euros = dollars * rate  
    println("€%.2f".format(euros))  
}
```

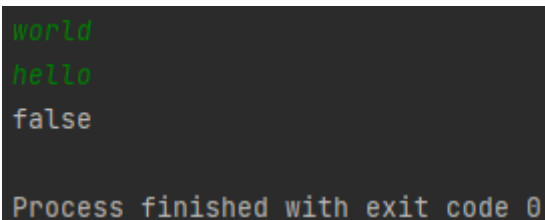


A terminal window showing the execution of the Kotlin program. The prompt "Введите сумму в долларах: " is followed by the input "2". The output is "€2,20". The terminal ends with "Process finished with exit code 0".

4.

```
fun isAnagram(str1: String, str2: String): Boolean {  
    return str1.lowercase().filter { it.isLetter() }.toList().sorted() ==  
        str2.lowercase().filter { it.isLetter() }.toList().sorted()  
}
```

```
fun main() {  
    val str1 = readLine() ?: ""  
    val str2 = readLine() ?: ""  
    println(isAnagram(str1, str2))  
}
```



A terminal window showing the execution of the Kotlin program. The input "world" is followed by "hello". The output is "false". The terminal ends with "Process finished with exit code 0".

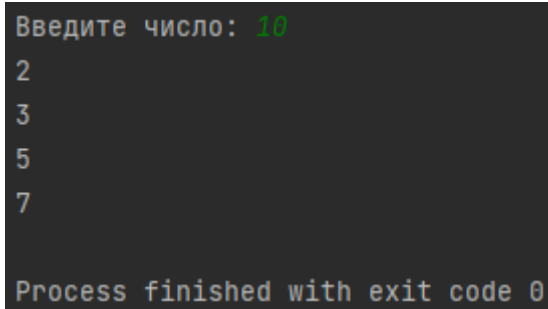
5.

```
fun isPrime(n: Int): Boolean {  
    if (n < 2) return false  
    return (2 until n).none { n % it == 0 }
```

```
}
```

```
fun findPrimes(n: Int) {  
    for (i in 2..n) {  
        if (isPrime(i)) println(i)  
    }  
}
```

```
fun main() {  
    print("Введите число: ")  
    val n = readLine()?.toIntOrNull() ?: 0  
    findPrimes(n)  
}
```



```
Введите число: 10  
2  
3  
5  
7  
  
Process finished with exit code 0
```

6.

```
fun sortStringsEfficient(inputArray: Array<String>): Array<String> {  
    return inputArray.sortedArray()  
}
```

```
fun main() {  
    println("Введите строки через пробел:")  
    val input = readLine() ?: ""  
  
    val stringArray = input.split(" ").toTypedArray()  
  
    if (stringArray.isNotEmpty() && stringArray[0].isNotEmpty()) {  
        println("\nИсходный массив:")  
    }
```

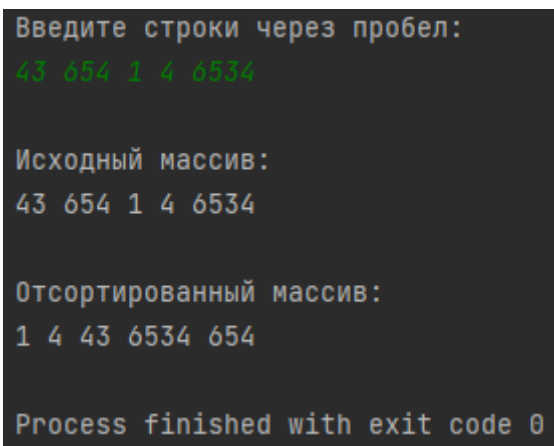
```

println(stringArray.joinToString(" "))

val sortedArray = sortStringsEfficient(stringArray)

println("\nОтсортированный массив:")
println(sortedArray.joinToString(" "))
} else {
    println("Массив пуст")
}
}
}

```



```

Введите строки через пробел:
43 654 1 4 6534

Исходный массив:
43 654 1 4 6534

Отсортированный массив:
1 4 43 6534 654

Process finished with exit code 0

```

7.

```

fun invertCase(text: String): String {
    // Проверка на пустую строку
    if (text.isEmpty()) return ""

    val result = StringBuilder()

    for (char in text) {
        val newChar = when {
            char.isUpperCase() -> char.lowercaseChar()
            char.isLowerCase() -> char.uppercaseChar()
            else -> char // Для не-буквенных символов оставляем как есть
        }
    }
}

```

```

        result.append(newChar)
    }

    return result.toString()
}

fun processText(text: String) {
    println("Исходный текст: $text")
    println("Результат: ${invertCase(text)}")
}

fun main() {
    println("Введите текст:")
    val input = readLine() ?: ""

    if (input.isNotEmpty()) {
        processText(input)
    } else {
        println("Текст не может быть пустым")
    }
}

```

```

Введите текст:
Russia
Исходный текст: Russia
Результат: rUSSIA

Process finished with exit code 0

```

8.

```

class NumberGuessingGame {
    private var targetNumber: Int = 0
    private var attempts: Int = 0
}

```

```
init {  
    targetNumber = (1..100).random()  
}
```

```
private fun readUserInput(): Int? {  
    print("Введите число (1-100): ")  
    return try {  
        readLine()?.toInt()?.also { number ->  
            if (number !in 1..100) {  
                println("Число должно быть от 1 до 100!")  
                return null  
            }  
        }  
    } catch (e: NumberFormatException) {  
        println("Пожалуйста, введите корректное число!")  
        null  
    }  
}
```

```
private fun checkGuess(guess: Int): Boolean {  
    attempts++  
  
    when {  
        guess < targetNumber -> {  
            println("Больше! (Попытка: $attempts)")  
            return false  
        }  
        guess > targetNumber -> {  
            println("Меньше! (Попытка: $attempts)")  
            return false  
        }  
        else -> {
```

```
        println("\nПоздравляем! Вы угадали число $targetNumber")
        println("Количество попыток: $attempts")
        return true
    }
}
}
```

```
fun play() {
    println("Добро пожаловать в игру 'Угадай число!'")
    println("Попробуйте угадать число от 1 до 100")
    println("Начинаем игру!\n")

    while (true) {
        val guess = readUserInput() ?: continue
        if (checkGuess(guess)) break
    }
}
}
```

```
fun main() {
    while (true) {
        val game = NumberGuessingGame()
        game.play()

        print("\nХотите сыграть еще раз? (да/нет): ")
        val playAgain = readLine()?.lowercase()

        if (playAgain != "да") {
            println("Спасибо за игру! До свидания!")
            break
        }
    }
    println()
}
```

```
}  
}
```

```
Добро пожаловать в игру 'Угадай число'!  
Попробуйте угадать число от 1 до 100  
Начинаем игру!  
  
Введите число (1-100): 50  
Меньше! (Попытка: 1)  
Введите число (1-100): 45  
Меньше! (Попытка: 2)  
Введите число (1-100): 35  
Больше! (Попытка: 3)  
Введите число (1-100): 40  
Больше! (Попытка: 4)  
Введите число (1-100): 42  
  
Поздравляем! Вы угадали число 42  
Количество попыток: 5  
  
Хотите сыграть еще раз? (да/нет):
```

9.

```
class PasswordGenerator {  
    private val lowercase = ('a'..'z').toList()  
    private val uppercase = ('A'..'Z').toList()  
    private val numbers = ('0'..'9').toList()  
    private val special = "!@#%&*()_+=[{}|;:,<>?".toList()  
  
    fun generatePassword(  
        length: Int = 12,  
        useLower: Boolean = true,  
        useUpper: Boolean = true,  
        useNumbers: Boolean = true,  
        useSpecial: Boolean = true  
    ): String {  
        // Создаем список доступных символов на основе параметров
```



```

val availableChars = mutableListOf<Char>()
if (useLower) availableChars.addAll(lowercase)
if (useUpper) availableChars.addAll(uppercase)
if (useNumbers) availableChars.addAll(numbers)
if (useSpecial) availableChars.addAll(special)

// Проверка на пустой набор символов
if (availableChars.isEmpty()) {
    return "Выберите хотя бы один набор символов"
}

// Генерация пароля
return buildString {
    repeat(length) {
        append(availableChars.random())
    }
}
}

fun generateMultiplePasswords(count: Int, length: Int): List<String> {
    return List(count) { generatePassword(length) }
}

fun main() {
    val generator = PasswordGenerator()

    println("Генератор паролей")
    println("=====")

    print("Введите длину пароля (по умолчанию 12): ")
    val length = readLine()?.toIntOrNull() ?: 12

```

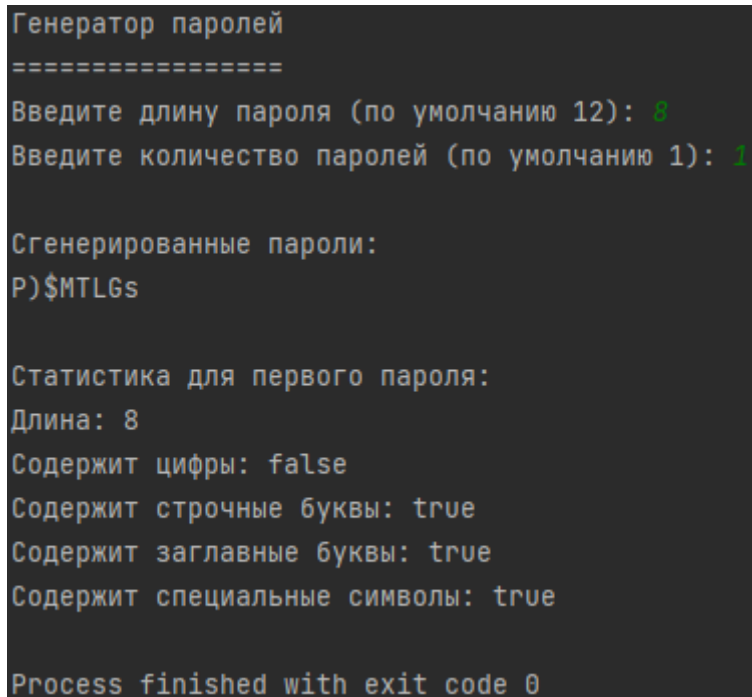
```

println("Введите количество паролей (по умолчанию 1): ")
val count = readLine()?.toIntOrNull() ?: 1

println("\nСгенерированные пароли:")
generator.generateMultiplePasswords(count, length).forEach { password ->
    println(password)
}

// Дополнительная статистика
val password = generator.generatePassword(length)
println("\nСтатистика для первого пароля:")
println("Длина: ${password.length}")
println("Содержит цифры: ${password.any { it.isDigit() }}")
println("Содержит строчные буквы: ${password.any { it.isLowerCase() }}")
println("Содержит заглавные буквы: ${password.any { it.isUpperCase() }}")
println("Содержит специальные символы: ${password.any { !it.isLetterOrDigit() }}")
}

```



```

Генератор паролей
=====
Введите длину пароля (по умолчанию 12): 8
Введите количество паролей (по умолчанию 1): 1

Сгенерированные пароли:
P)$MTLGs

Статистика для первого пароля:
Длина: 8
Содержит цифры: false
Содержит строчные буквы: true
Содержит заглавные буквы: true
Содержит специальные символы: true

Process finished with exit code 0

```

```

class TextAnalyzer {
    fun findLongestWord(text: String): String {
        // Разбиваем текст на слова, удаляя знаки препинания и лишние пробелы
        val words = text.split(Regex("[ ,!?:;\\\"()\\n\\t]+"))
            .filter { it.isNotEmpty() }

        if (words.isEmpty()) return "Текст не содержит слов"

        // Находим самое длинное слово и все слова такой же длины
        val maxLength = words.maxOf { it.length }
        val longestWords = words.filter { it.length == maxLength }

        return buildString {
            append("Самое длинное слово: ${longestWords[0]} (длина: $maxLength символов)")
            if (longestWords.size > 1) {
                append("\\nДругие слова такой же длины: ")
                append(longestWords.drop(1).joinToString(", "))
            }
        }
    }

    fun getTextStatistics(text: String): String {
        if (text.isEmpty()) return "Текст пуст"

        val words = text.split(Regex("[ ,!?:;\\\"()\\n\\t]+")).filter { it.isNotEmpty() }
        val sentences = text.split(Regex("[.!?]+")).filter { it.isNotEmpty() }

        return buildString {
            append("Статистика текста:")
            append("\\n- Количество символов: ${text.length}")
            append("\\n- Количество символов без пробелов: ${text.count { !it.isWhitespace() }}")
            append("\\n- Количество слов: ${words.size}")
        }
    }
}

```

```

        append("\n- Количество предложений: ${sentences.size}")

        append("\n- Средняя длина слова: ${"%0.1f".format(words.sumOf { it.length }.toDouble() /
words.size)}")

        append("\n\n${findLongestWord(text)}")
    }
}
}

```

```

fun main() {

    val analyzer = TextAnalyzer()

    println("Введите текст (для завершения введите пустую строку):")

    val textBuilder = StringBuilder()

    while (true) {
        val line = readLine() ?: break
        if (line.isEmpty()) break
        textBuilder.append(line).append("\n")
    }

    val text = textBuilder.toString().trim()

    if (text.isNotEmpty()) {
        println("\n" + analyzer.getTextStatistics(text))
    } else {
        println("Текст не был введен")
    }
}

```

Введите текст (для завершения введите пустую строку):

I'm a Steve

Статистика текста:

- Количество символов: 11
- Количество символов без пробелов: 9
- Количество слов: 4
- Количество предложений: 1
- Средняя длина слова: 2,0

Самое длинное слово: Steve (длина: 5 символов)

Process finished with exit code 0

