Творческое задание

Задание 1

import kotlin.random.Random  
  
fun main() {  
 *println*("10 случайных чисел от 1 до 100:")  
  
 for (i in 1..10) {  
 val randomNumber = Random.nextInt(1, 101) // Генерация случайного числа от 1 (включительно) до 101 (не включительно), т.е. от 1 до 100  
 *println*(randomNumber)  
 }  
}

Ответ:

10 случайных чисел от 1 до 100:

43

89

26

48

11

71

54

78

36

26

Process finished with exit code 0

Задание 2

fun analyzeString(str: String) {  
 val vowels = "aeiouAEIOU"  
 var vowelCount = 0  
 var consonantCount = 0  
  
 for (char in str) {  
 if (char.*isLetter*()) {  
 if (vowels.*contains*(char)) {  
 vowelCount++  
 } else {  
 consonantCount++  
 }  
 }  
 }  
  
 *println*("Строка: $str")  
 *println*("Гласных: $vowelCount")  
 *println*("Согласных: $consonantCount")  
}  
  
fun main() {  
 val inputString = "Hello, Kotlin!"  
 *analyzeString*(inputString)  
  
 val anotherString = "This is a sample string with vowels and consonants."  
 *analyzeString*(anotherString)  
  
 val stringWithNumbers = "123 abc 456 def"  
 *analyzeString*(stringWithNumbers) //Проверка обработки строк с цифрами.  
  
 val emptyString = ""  
 *analyzeString*(emptyString) //Проверка обработки пустой строки  
}

Ответ:

Строка: Hello, Kotlin!

Гласных: 4

Согласных: 7

Строка: This is a sample string with vowels and consonants.

Гласных: 13

Согласных: 29

Строка: 123 abc 456 def

Гласных: 2

Согласных: 4

Строка:

Гласных: 0

Согласных: 0

Process finished with exit code 0

Задание 3

import java.util.\*  
  
fun main() {  
 val scanner = Scanner(System.*`in`*)  
  
 // Курсы валют (примеры, необходимо обновлять реальные значения)  
 val exchangeRates = *mapOf*(  
 "USD" *to mapOf*("EUR" *to* 0.92, "GBP" *to* 0.80, "RUB" *to* 90.0),  
 "EUR" *to mapOf*("USD" *to* 1.09, "GBP" *to* 0.87, "RUB" *to* 98.0),  
 "GBP" *to mapOf*("USD" *to* 1.25, "EUR" *to* 1.15, "RUB" *to* 112.0),  
 "RUB" *to mapOf*("USD" *to* 0.011, "EUR" *to* 0.010, "GBP" *to* 0.0089)  
 )  
  
 // Получаем входные данные от пользователя  
 *print*("Введите сумму для конвертации: ")  
 val amount = scanner.nextDouble()  
  
 *print*("Введите исходную валюту (USD, EUR, GBP, RUB): ")  
 val fromCurrency = scanner.next().*toUpperCase*()  
  
 *print*("Введите целевую валюту (USD, EUR, GBP, RUB): ")  
 val toCurrency = scanner.next().*toUpperCase*()  
  
 // Проверяем валидность валют  
 if (!exchangeRates.containsKey(fromCurrency) || !exchangeRates.containsKey(toCurrency)) {  
 *println*("Ошибка: Неверная валюта.")  
 return  
 }  
  
 // Проверяем наличие курса обмена  
 if (!exchangeRates[fromCurrency]!!.containsKey(toCurrency)) {  
 *println*("Ошибка: Нет курса обмена для этих валют.")  
 return  
 }  
  
 // Выполняем конвертацию  
 val exchangeRate = exchangeRates[fromCurrency]!![toCurrency]!!  
 val convertedAmount = amount \* exchangeRate  
  
 // Выводим результат  
 *println*("$amount $fromCurrency = ${"%.2f".*format*(convertedAmount)} $toCurrency")  
}

Ответ:

Введите сумму для конвертации: 65

Введите исходную валюту (USD, EUR, GBP, RUB): USD

Введите целевую валюту (USD, EUR, GBP, RUB): RUB

65.0 USD = 5850,00 RUB

Process finished with exit code 0

Задание 4

fun являютсяАнаграммами(строка1: String, строка2: String): Boolean {  
 // 1. Удаляем пробелы и приводим к нижнему регистру для нормализации  
 val s1 = строка1.*replace*(" ", "").*toLowerCase*()  
 val s2 = строка2.*replace*(" ", "").*toLowerCase*()  
  
 // 2. Если длины разные, они не могут быть анаграммами  
 if (s1.length != s2.length) {  
 return false  
 }  
  
 // 3. Сортируем обе строки и сравниваем  
 return s1.*toCharArray*().*sorted*().*joinToString*("") == s2.*toCharArray*().*sorted*().*joinToString*("")  
  
 // Альтернативные решения (с использованием HashMap или массивов char для подсчета частот) см. ниже  
}  
  
fun main() {  
 *println*(" 'listen' и 'silent' анаграммы: ${*являютсяАнаграммами*("listen", "silent")}") // true  
 *println*(" 'hello' и 'world' анаграммы: ${*являютсяАнаграммами*("hello", "world")}") // false  
 *println*(" 'Dormitory' и 'dirty room' анаграммы: ${*являютсяАнаграммами*("Dormitory", "dirty room")}") // true  
 *println*(" 'Conversation' и 'voices rant on' анаграммы: ${*являютсяАнаграммами*("Conversation", "voices rant on")}") // true  
 *println*(" 'Programming' и 'граммирование' анаграммы: ${*являютсяАнаграммами*("Programming", "граммирование")}") // false (разные языки)  
}

Ответ:

'listen' и 'silent' анаграммы: true

'hello' и 'world' анаграммы: false

'Dormitory' и 'dirty room' анаграммы: true

'Conversation' и 'voices rant on' анаграммы: true

'Programming' и 'граммирование' анаграммы: false

Process finished with exit code 0

Задание 5

fun main() {  
 *print*("Введите число N: ")  
 val n = *readLine*()?.*toIntOrNull*()  
  
 if (n == null || n <= 1) {  
 *println*("Пожалуйста, введите целое число больше 1.")  
 return  
 }  
  
 *println*("Простые числа до $n:")  
 *findPrimes*(n).*forEach* **{** prime **->** *print*("$prime ")  
 **}** *println*()  
}  
  
fun findPrimes(n: Int): List<Int> {  
 val primes = *mutableListOf*<Int>()  
 if (n >= 2) {  
 primes.add(2) // 2 - первое простое число  
 }  
  
 for (i in 3..n *step* 2) { // Проверяем только нечетные числа, начиная с 3  
 if (*isPrime*(i, primes)) {  
 primes.add(i)  
 }  
 }  
  
 return primes  
}  
  
fun isPrime(number: Int, knownPrimes: List<Int>): Boolean {  
 //Оптимизация: проверяем делимость только до корня из числа  
 val sqrtNumber = kotlin.math.*sqrt*(number.toDouble()).toInt()  
  
 //Оптимизация: проверяем делимость только на известные простые числа,  
 // и останавливаемся, если текущий простой делитель больше корня из number  
 for (prime in knownPrimes) {  
 if (prime > sqrtNumber) {  
 break  
 }  
 if (number % prime == 0) {  
 return false  
 }  
 }  
 return true  
}

Ответ:

Введите число N: 55

Простые числа до 55:

2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53

Process finished with exit code 0

Задание 6

fun sortStringsAlphabetically(strings: Array<String>): Array<String> {  
 return strings.*sortedArray*()  
}  
  
// Пример использования:  
fun main() {  
 val unsortedArray = *arrayOf*("banana", "apple", "orange", "grape")  
 val sortedArray = *sortStringsAlphabetically*(unsortedArray)  
  
 *println*("Несортированный массив: ${unsortedArray.*contentToString*()}") // Вывод: Несортированный массив: [banana, apple, orange, grape]  
 *println*("Сортированный массив: ${sortedArray.*contentToString*()}") // Вывод: Сортированный массив: [apple, banana, grape, orange]  
}

Ответ:

Несортированный массив: [banana, apple, orange, grape]

Сортированный массив: [apple, banana, grape, orange]

Process finished with exit code 0

Задание 7

fun main() {  
 val inputString = "Hello World!"  
 val resultString = *changeCase*(inputString)  
 *println*("Исходная строка: $inputString")  
 *println*("Строка с измененным регистром: $resultString")  
}  
  
fun changeCase(inputString: String): String {  
 val result = StringBuilder()  
 for (char in inputString) {  
 when {  
 char.*isUpperCase*() -> result.append(char.*toLowerCase*())  
 char.*isLowerCase*() -> result.append(char.*toUpperCase*())  
 else -> result.append(char) // Оставляем символы, не являющиеся буквами, без изменений  
 }  
 }  
 return result.toString()  
}

Ответ:

Исходная строка: Hello World!

Строка с измененным регистром: hELLO wORLD!

Process finished with exit code 0

Задание 8

import kotlin.random.Random  
  
fun main() {  
 val randomNumber = Random.nextInt(1, 101) // Генерируем случайное число от 1 до 100  
 var attempts = 0  
  
 *println*("Добро пожаловать в игру 'Угадай число'!")  
 *println*("Я загадал число от 1 до 100. Попробуйте угадать.")  
  
 do {  
 *print*("Введите ваше предположение: ")  
 val guess = *readLine*()?.*toIntOrNull*()  
  
 if (guess == null) {  
 *println*("Некорректный ввод. Пожалуйста, введите число.")  
 continue // Переходим к следующей итерации цикла  
 }  
  
 attempts++  
  
 when {  
 guess < randomNumber -> *println*("Загаданное число больше.")  
 guess > randomNumber -> *println*("Загаданное число меньше.")  
 else -> {  
 *println*("Поздравляю! Вы угадали число $randomNumber за $attempts попыток.")  
 break // Выходим из цикла, если число угадано  
 }  
 }  
 } while (true)  
}

Ответ:

Добро пожаловать в игру 'Угадай число'!

Я загадал число от 1 до 100. Попробуйте угадать.

Введите ваше предположение: 5

Загаданное число больше.

Введите ваше предположение: 50

Загаданное число больше.

Введите ваше предположение: 60

Загаданное число больше.

Введите ваше предположение: 80

Загаданное число меньше.

Введите ваше предположение: 77

Загаданное число меньше.

Введите ваше предположение: 75

Загаданное число меньше.

Введите ваше предположение: 73

Поздравляю! Вы угадали число 73 за 7 попыток.

Process finished with exit code 0

Задание 9

import kotlin.random.Random  
  
fun generatePassword(length: Int): String {  
 val charset = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789!@#$%^&\*()\_+=-`~[]{}|;':\",./<>?"  
 return (1..length)  
 .*map* **{** charset.*random*(Random) **}** .*joinToString*("")  
}  
  
  
fun main() {  
 val password = *generatePassword*(12)  
 *println*(password) // Пример: kY2%jT8!aL#  
}

Ответ:

R]l|c)1)Sa8s

Process finished with exit code 0

Задание 10

import java.lang.IllegalArgumentException  
  
fun findLongestWord(text: String): String {  
 if (text.*isBlank*()) {  
 throw IllegalArgumentException("Входная строка не может быть пустой или содержать только пробелы.")  
 }  
  
 val words = text.*split*(Regex("\s+")) // Разделяем строку на слова по одному или нескольким пробелам  
 .*filter* **{ it**.*isNotBlank*() **}** // Убираем пустые строки, которые могли появиться из-за лишних пробелов  
 .*map* **{ it**.*replace*(Regex("[^a-zA-Z0-9]"), "") **}** // Убираем знаки препинания  
  
 var longestWord = ""  
 for (word in words) {  
 if (word.length > longestWord.length) {  
 longestWord = word  
 }  
 }  
  
 return longestWord  
}  
  
  
fun main() {  
 val text1 = "fun main() { println(\"Hello, world!\") }"  
 *println*(*findLongestWord*(text1)) // Вывод: println  
  
 val text2 = "This is a test string with some long words like extraordinarily and unquestionably."  
 *println*(*findLongestWord*(text2)) // Вывод: extraordinarily  
  
 val text3 = " with some spaces "  
 *println*(*findLongestWord*(text3)) // Вывод: with  
  
  
 val emptyText = ""  
 try {  
 *println*(*findLongestWord*(emptyText))  
 } catch (e: IllegalArgumentException) {  
 *println*(e.message) // Вывод: Входная строка не может быть пустой или содержать только пробелы.  
 }  
  
  
 val blankText = " "  
 try {  
 *println*(*findLongestWord*(blankText))  
 } catch (e: IllegalArgumentException) {  
 *println*(e.message) // Вывод: Входная строка не может быть пустой или содержать только пробелы.  
 }  
}

Ответ:

funmainprintlnHelloworld

likeextraordinarilyandunque

with

Входная строка не может быть пустой или содержать только пробелы.

Входная строка не может быть пустой или содержать только пробелы.

Process finished with exit code 0