Вариант 16

1. минимальную цифру, некратную трем

fun main() {  
 *print*("Введите неотрицательное целое число: ")  
 val input = *readLine*()?.*toIntOrNull*()  
 if (input == null || input < 0)  
 {  
 *println*("ошибка")  
 return  
 }  
 var minDigit = 9  
 for (digit in input.toString()) {  
 var digitInt = digit.*digitToInt*()  
 if (digitInt % 3 == 0)  
 continue  
 if (digitInt >= minDigit)  
 continue  
 minDigit = digitInt  
 }  
 if (minDigit == 9)  
 return  
 *println*("Минимальная цифра, некратная трем: $minDigit")  
}

(+)

2) В строке указано несколько слов, разделенных пробелами (по одному пробелу между словами). Какое количество слов удовлетворяет условию наличия повторяющихся символов? Предполагается, что все символы в строке имеют код, не больший, чем 127.

fun main(args: Array<String>) {  
 val str = "aaa bbb gaga "  
 *println*("Количество слов с повторяющимися символами: ${*getAnswer*(str)}")  
}  
  
fun getAnswer(str: String): Int {  
 // if (str.isEmty())  
 // return 0  
 return *getCountDuplicatedWord*(str)  
}  
  
fun getCountDuplicatedWord(str: String): Int {  
 var count = 0  
 var hasDuplicates = false  
 var isCharacterInWord = BooleanArray(128)  
  
 for (i in 0..str.length-1)  
 {  
 if (str[i].*isWhitespace*() || str[i] == '.') {  
 isCharacterInWord = BooleanArray(128)  
 hasDuplicates = false  
 continue  
 }  
  
 if (hasDuplicates)  
 continue  
  
 if (isCharacterInWord[str[i].*code*]) {  
 hasDuplicates = true  
 count += 1  
 continue  
 }  
  
 isCharacterInWord[str[i].*code*] = true  
 }  
 return count  
}

1. Найдите последний символ в последнем самом коротком слове в строке с четным числом символов (в строке указываются только слова, разделенные одним или несколькими пробелами).

fun main(args: Array<String>)  
{  
 val str = "11 2222"

if (str.*isEmpty*())  
 {  
 *println*("Ответ: Строка пустая")  
 return  
 }  
  
 if(*findLastChar*(str) == ' ')  
 {  
 *println*("Нет ответа")  
 return  
 }  
 else  
 *println*("Последний символ в последнем самом коротком слове\n" +  
 " с чётным числом символов: ${*findLastChar*(str)}") //чётное слово  
}  
fun findLastChar(str: String): Char  
{  
 var size = 0  
 var oldSize = Int.MAX\_VALUE  
 var res = ' '  
 for (i in 0 *until* str.length)  
 {  
 if (str[i].*isWhitespace*() || str[i] == '.')  
 {  
 if (*isEven*(size) && size <= oldSize)  
 {  
 res = str[i - 1]  
 oldSize = size  
 }  
 size = 0  
 }  
 else  
 size++  
 }  
 if (*isEven*(size) && size <= oldSize)  
 {  
 res = str[str.length - 1]  
 oldSize = size  
 }  
 return res  
}  
fun isEven(num: Int): Boolean = (num % 2 == 0)

(показать)

Методы расширения println()

1. Map - Возвращает список, содержащий результаты применения данной функции преобразования к каждому символу в исходной последовательности символов.
2. forEach - Выполняет заданное действие над каждым элементом.
3. Fold - Накапливает значение, начиная с начального значения и применяя операцию слева направо к текущему значению аккумулятора и каждому символу.

Возвращает указанное начальное значение, если последовательность символов пуста.

Параметры:

операция - функция, которая принимает текущее значение аккумулятора и символ и вычисляет следующее значение аккумулятора.

1. Reduce - Аккумулирует значение, начиная с первого символа, и применяет операцию слева направо к текущему значению аккумулятора и каждому символу.

Выдает исключение, если эта последовательность символов пуста. Если последовательность символов может быть пустой ожидаемым образом, используйте вместо этого сокращениеOrNull. Он возвращает значение null, когда его получатель пуст.

Параметры:

операция - функция, которая принимает текущее значение аккумулятора и символ и вычисляет следующее значение аккумулятора.

Образцы:

образцы.коллекции.Коллекции.Агрегаты.редуце

1. foldRight - Накапливает значение, начиная с начального значения и применяя операцию справа налево к каждому символу и текущему значению аккумулятора.

Возвращает указанное начальное значение, если последовательность символов пуста.

Параметры:

операция — функция, которая принимает символ и текущее значение аккумулятора и вычисляет следующее значение аккумулятора.

1. Reverced - Возвращает строку с символами в обратном порядке.
2. Any - Возвращает строку с символами в обратном порядке.
3. All - возвращает true, если все символы соответствуют данному предикату.
4. Обратите внимание: если последовательность символов не содержит символов, функция возвращает true, поскольку в ней нет символов, не соответствующих предикату. Более подробное объяснение этой логической концепции смотрите в статье «Пустая истина».
5. None - Возвращает true, если последовательность символов не содержит символов.
6. takeWhile - Возвращает строку, содержащую первые символы, удовлетворяющие заданному предикату.
7. generateSequence (при приравниение (не через точку) - возвращает последовательность, которая вызывает функцию для вычисления следующего значения на каждой итерации, пока функция не вернет ноль.

Возвращенная последовательность ограничена возможностью повторения только один раз.

val numbers = generateSequence{ number += 2; number}

1. last – возвращает последний элемент

*println*("a".*last*() )

1. toSet - Возвращает набор всех символов.

Возвращенный набор сохраняет порядок итерации элементов исходной последовательности символов.

1. Sorted – сортировка по возрастанию (val a = b.sorted())
2. sortedBy – сортировка по аргументу(it) по возрастанию

var a = b.sortedBy{it.(поле объекта)}

1. sortedByDescending - сортировка по аргументу(it) по убыванию

var a = b.sortedByDescending{it.(поле объекта)}

1. joinToString – помещает в строку выбранные элементы (можно числа)

val numbers = listOf(1, 2, 3, 4, 5, 6)

println(numbers.joinToString()) // 1, 2, 3, 4, 5, 6

1. lines - Разбивает эту последовательность символов на список строк, разделенных любой из следующих последовательностей символов: CRLF, LF или CR. Возвращенные строки не содержат завершающих разделителей строк.

*println*("a".*lines*())

1. average - получение среднего значения

val numbers = listOf(4, 6, 3, 5, 1, 2)

val avg = numbers.average()

println(avg)    // 3.5

1. filter - Возвращает строку, содержащую только те символы исходной строки, которые соответствуют заданному предикату.

people = people.filter{it.length == 3}    // получаем значения с длиной в 3 символа

1. groupBy – группировка по ключ/значение

val words = listOf("a", "abc", "ab", "def", "abcd")

val byLength = words.groupBy { it.length }

println(byLength.keys) // [1, 3, 2, 4]

println(byLength.values) // [[a], [abc, def], [ab], [abcd]]

val mutableByLength: MutableMap<Int, MutableList<String>> = words.groupByTo(mutableMapOf()) { it.length }

// same content as in byLength map, but the map is mutable

println("mutableByLength == byLength is ${mutableByLength == byLength}") // true

1. groupingBy – groupBy и затем операция ко всем элементам

Если вам требуется сгруппировать элементы, а затем применить какую-либо операцию ко всем группам одновременно, то используйте функцию [groupingBy()](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.collections/grouping-by.html). Она возвращает экземпляр типа [Grouping](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.collections/-grouping/index.html). В свою очередь Grouping позволяет "лениво" применять операции ко всем группам: фактически группы будут создаваться прямо перед выполнением операции.

Grouping поддерживает следующие операции:

* [eachCount()](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.collections/each-count.html) - подсчитывает количество элементов в каждой группе.
* [fold()](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.collections/fold.html) и [reduce()](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.collections/reduce.html) - выполняют операции [fold и reduce](https://kotlinlang.ru/docs/reference/collection-aggregate.html#fold-and-reduce) для каждой группы как для отдельной коллекции, после чего возвращают результат.
* [aggregate()](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.collections/aggregate.html) - последовательно применяет данную операцию ко всем элементам в каждой группе, после чего возвращает результат. Это общий способ выполнения любых операций с объектом Grouping. Используйте его для реализации собственных операций, когда операций fold и reduce недостаточно.

**fun** main() {

**val** numbers = listOf(**"one"**, **"two"**, **"three"**, **"four"**, **"five"**, **"six"**)

println(numbers.groupingBy { it.first() }.eachCount()) *// {o=1, t=2, f=2, s=1}*

}

1. Intersect -  Функция возвращает набор, содержащий все элементы, содержащиеся как в первом, так и во втором списке

fun main() {

    val first = listOf(1, 3, 1, 6, 5, 7)

    val second = listOf(2, 3, 4, 5)

    val common = first.intersect(second)

    println(common)        // [3, 5]

}

1. Union – Возвращает набор, содержащий все отдельные элементы из обеих коллекций. Возвращенный набор сохраняет порядок итерации элементов исходного массива. Те элементы [другой](https://runebook.dev/ru/docs/kotlin/api/latest/jvm/stdlib/kotlin.collections/union#kotlin.collections%24union(kotlin.Array((kotlin.collections.union.T)),%20kotlin.collections.Iterable((kotlin.collections.union.T)))/other) коллекции, которые являются уникальными, повторяются в конце в порядке, установленном в [другой](https://runebook.dev/ru/docs/kotlin/api/latest/jvm/stdlib/kotlin.collections/union#kotlin.collections%24union(kotlin.Array((kotlin.collections.union.T)),%20kotlin.collections.Iterable((kotlin.collections.union.T)))/other) коллекции.

val oneToSix = listOf(1, 2, 3, 4, 5, 6)

val fourToTen = listOf(4, 5, 6, 7, 8, 9, 10)

val oneToTen = oneToSix.union(fourToTen)

assertThat(oneToTen).containsExactly(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)

1. Substract - Возвращает набор,содержащий все элементы,которые содержатся в данной коллекции и не содержатся в указанной коллекции. Возвращаемый комплект сохраняет порядок итераций элементов оригинальной коллекции.
2. Drop - получения всех элементов, кроме указанного количества первых или последних элементов, используйте функции [drop()](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.collections/drop.html) и [dropLast()](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.collections/drop-last.html)

fun main()

{

val numbers = listOf("one", "two", "three", "four", "five", "six")

println(numbers.drop(1)) // [two, three, four, five, six]

}

1. dropWhile - удаляет из потока элементы, пока они не начнут соответствовать некоторому условию:

peopleFlow.dropWhile{ person -> person.age > 17}

.collect { person -> println("name: ${person.name}   age:  ${person.age} ")}

…

1. partition - partition() функция для разделения списка на два, где первый список содержит элементы, для которых возвращался указанный предикат true, а второй список содержит оставшиеся элементы, для которых предикат вернул false.
2. Chunked – разделение коллекции по заданному количеству
3. Windowed -  все возможные диапазоны элементов коллекции с количеством для списка
4. Count -  количество объектов в потоке
5. Zip - берёт два списка и создаёт из их элементов пары
6. fun main(args: Array<String>) {  
    val str = "aaa bbb aga "  
    *println*("Количество слов с повторяющимися символами: ${*getAnswer*(str)}")  
   }  
     
   fun getAnswer(str: String): Int {  
    // if (str.isEmty())  
    // return 0  
    return *getCountDuplicatedWord*(str)  
   }  
     
   fun getCountDuplicatedWord(str: String): Int {  
    var size = 0  
    var countLocal = 0  
    var countGlobal = 0  
    var locationBegin = 0  
    var locationEnd = 0  
    //разделение на слова  
    for (i in 1.. str.length - 1)  
    {  
    if (!str[i].*isWhitespace*() && str[i] != '.')  
    {  
    if(size == 0)  
    locationBegin = i  
    size += 1  
    }  
    //обработка слова  
    else  
    {  
    locationEnd = i  
    for (x in locationBegin *until* locationEnd)  
    {  
    if(str[x - 1] == str[x]){  
    if(countLocal >= 1)  
    break  
    countLocal++;  
    }  
    else if(str[x - 1] != str[x] && countLocal >= 1){  
    countLocal = 0;  
    }  
    }  
    if (countLocal >= 1)  
    countGlobal++  
    countLocal = 0  
    size = 0  
    //обработка слова  
    }  
    }  
    return countGlobal  
   }