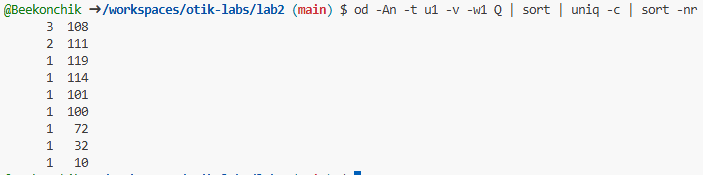


1 строка, 2 слова, 12 байтов





Преобразовываем каждый байт файла в его числовое значение 0-255 и выводим каждое значение на отдельной строчке, сортируем, подсчитываем количество каждого уникального байта, сортируем по убыванию.

3 раза символ “l” (код 108)

2 раза символ “o” (код 111)

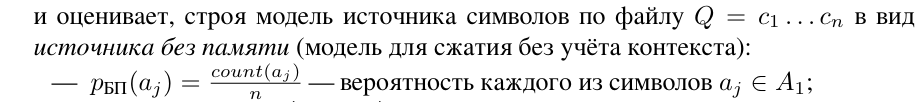
1 раз символ “w” (код 119)

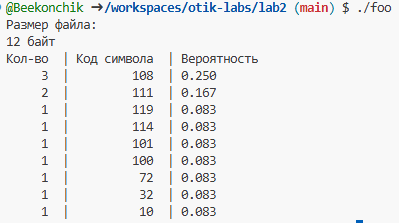
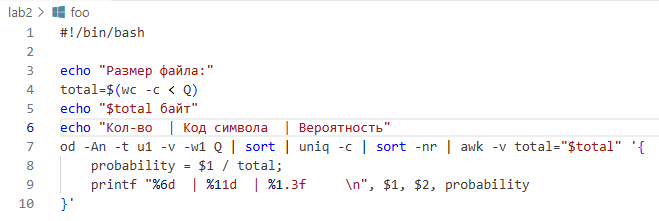
…

1 раз символ “ “ (код 32)

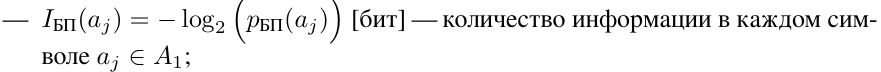
1 символ “\n” (код 10)

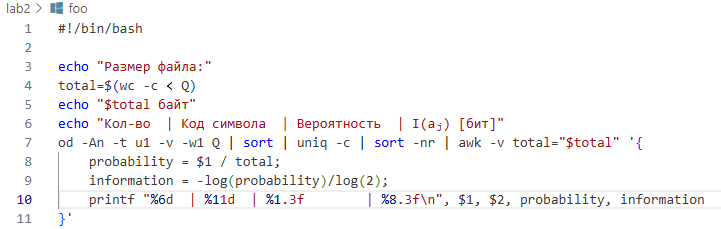
Изначальное содержание — “Hello world”

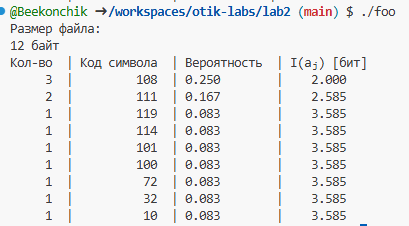




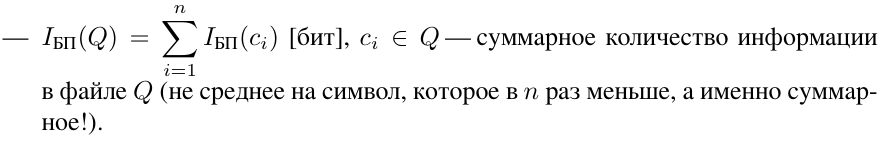
Добавляем все использованные раннее скрипты-однострочники в один файл foo. Присваиваем переменной total размер файла Q. Передаем из bash в awk total. Считаем вероятность вхождения каждого символа probability через . Первое поле $1 — общее количество вхождения символа в файл, второе поле $2 — код символа.

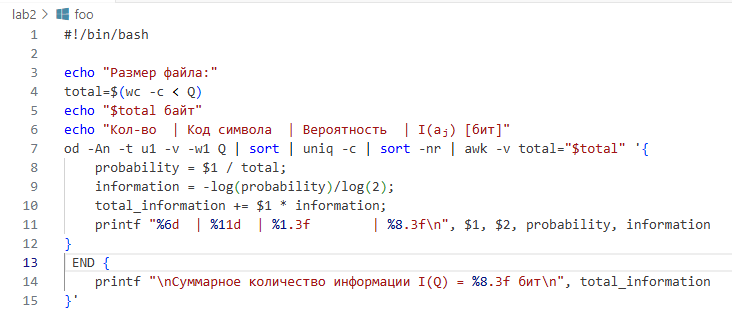






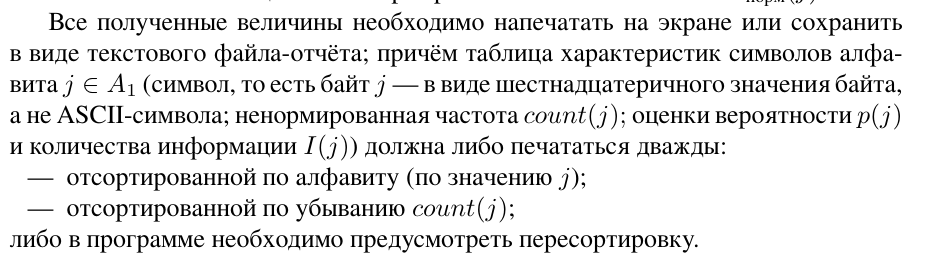
Считаем количество информации в каждом символе information через , добавляем вывод переменной.

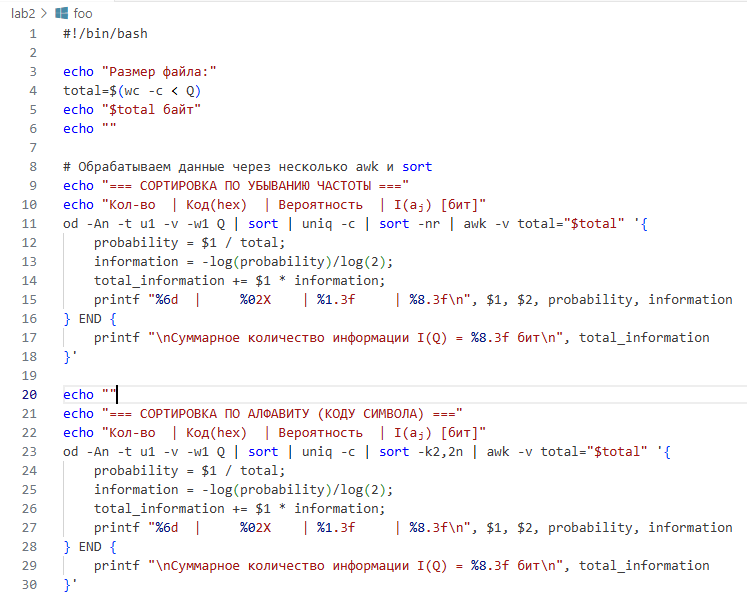
****

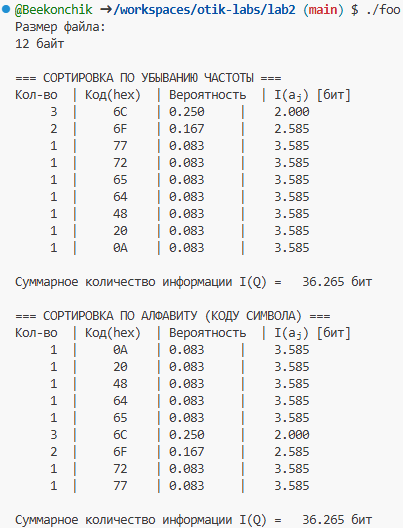
****

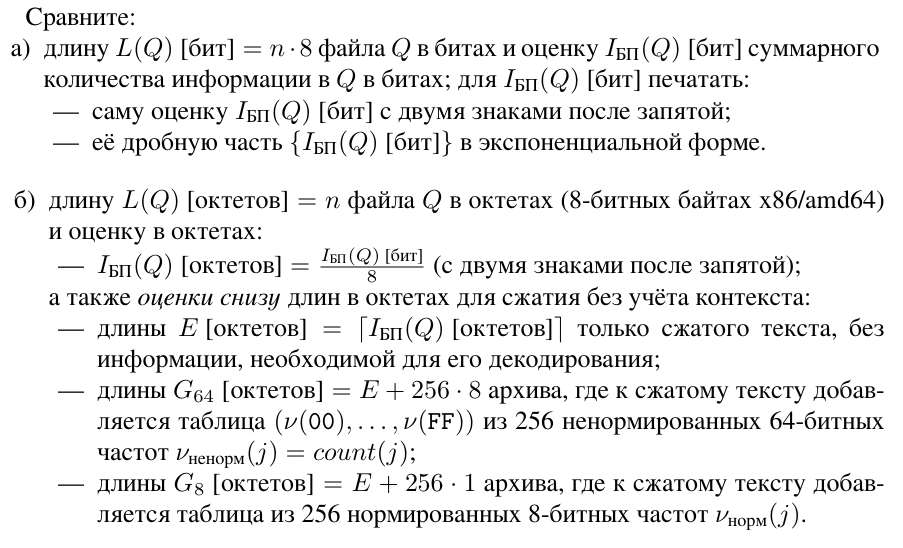
****

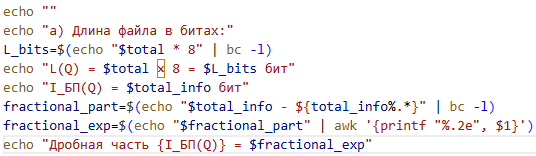
Считаем суммарное количество информации в файле, добавляя к сумме количество информации в каждом символе, умноженное на количество вхождений этого символа в файл (напр., количество информации в символе “l” с кодом 108, умноженное на 3).

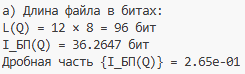




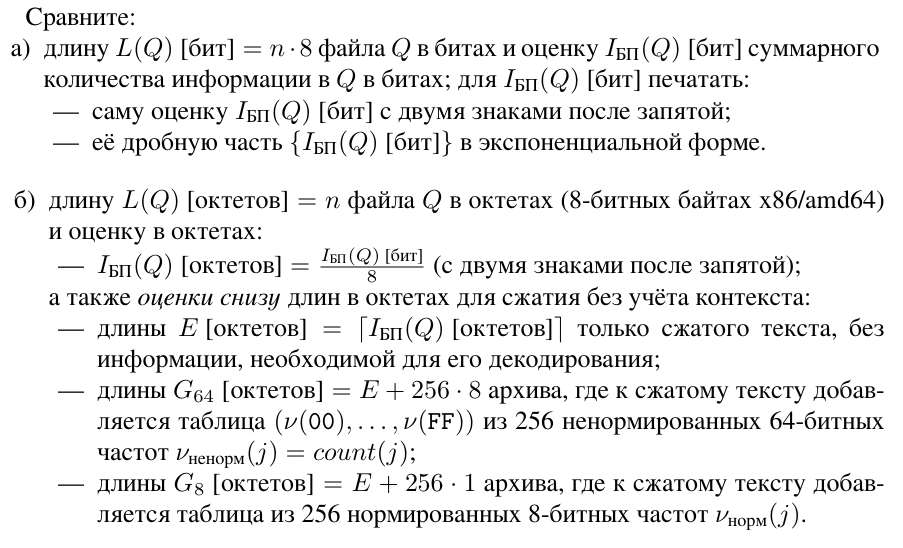


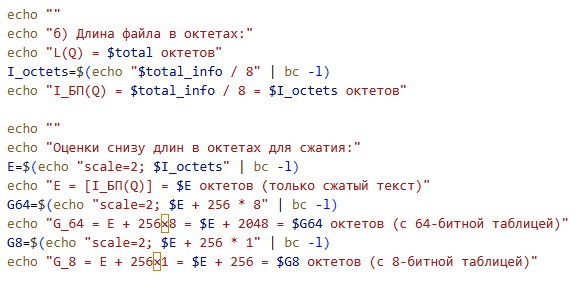


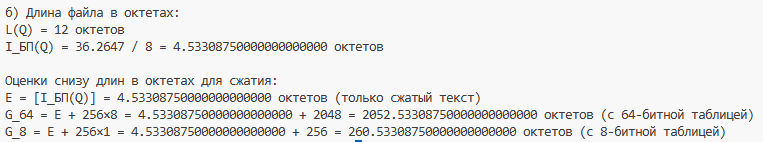




То есть файл избыточен, можно ужать файл от 96 бит до 37 бит.

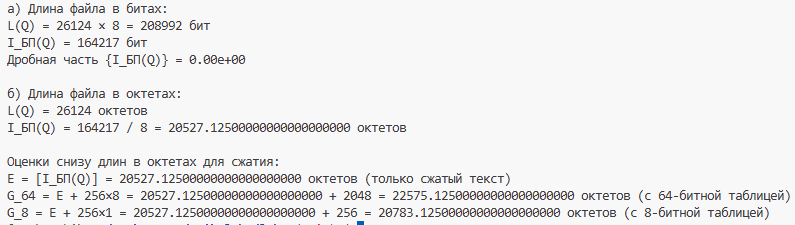




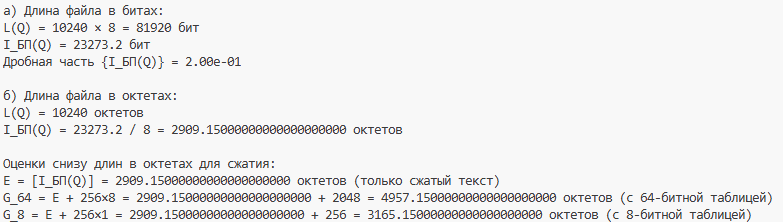


То есть сжимать файл неэффективно, поскольку размер файла со сжатым текстом и таблицей для декодирования будет превышать размер изначального файла.

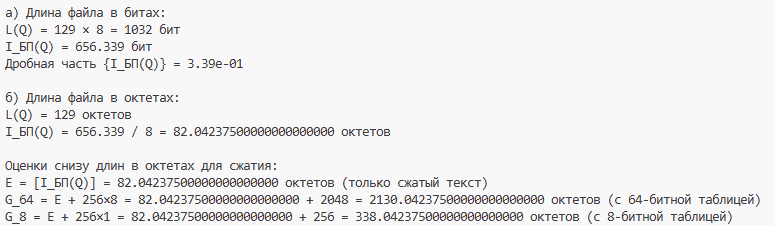
Для файла Файлы в разных форматах/hexdump:



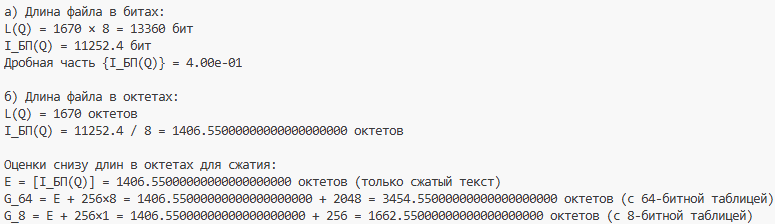
Для файла Файлы в разных форматах/test.doc:

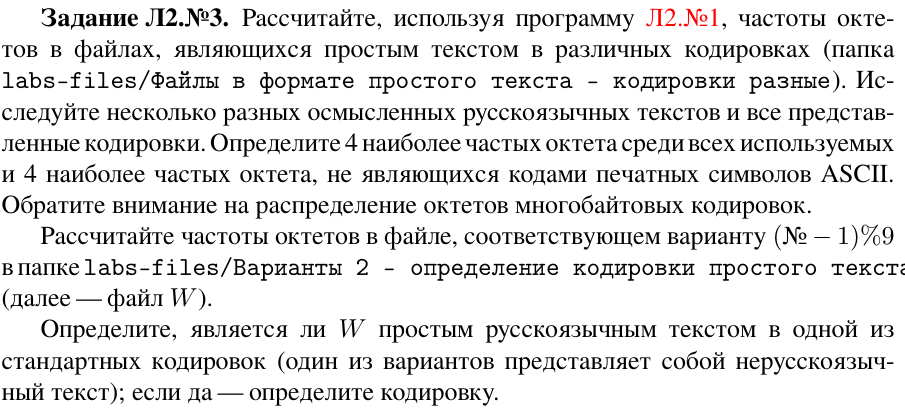


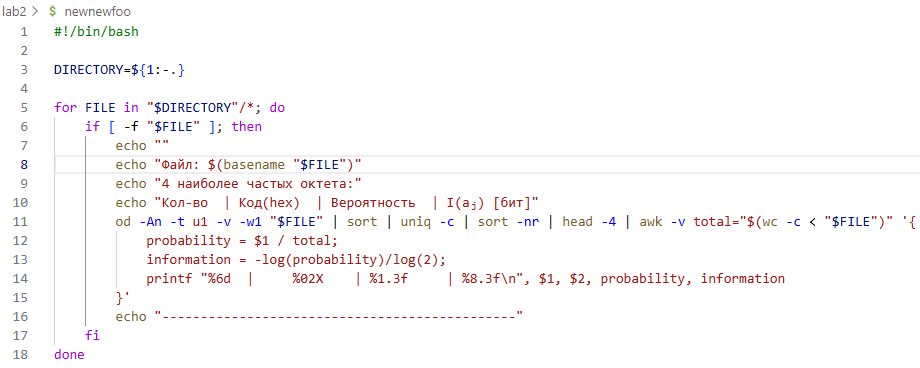
Для файла Файлы в разных форматах/chess16.png :



Для файла Файлы в разных форматах/beep.mp3 :







**4 наиболее частых октета:**

00 («нулевой символ») встречается в многобайтовых кодировках и является управляющим символом ASCII

04 (значение зависит от соседних октетов) встречается в многобайтовых кодировках и является управляющим символм ASCII

20 («пробел») встречается и в многобайтовых, и в однобайтовых кодировках, является печатным символом ASCII

D0 (значение зависит от соседних октетов) встречается в многобайтовых кодировках и является управляющим символом ASCII

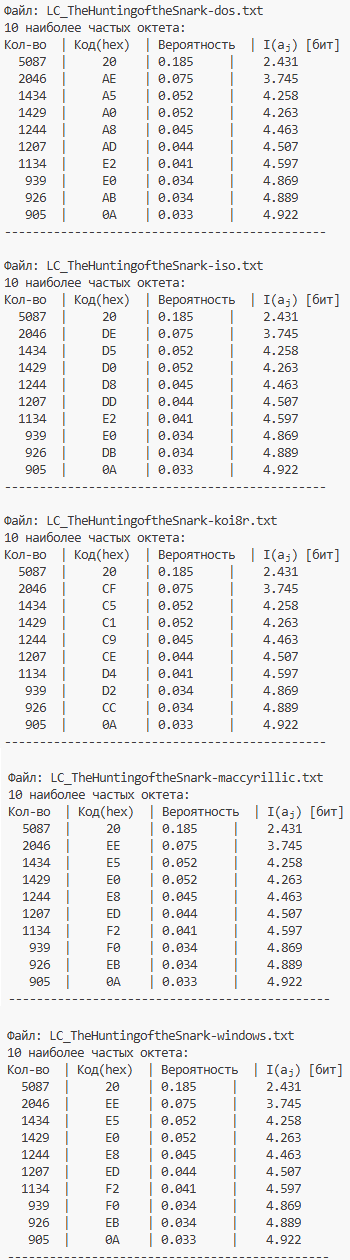
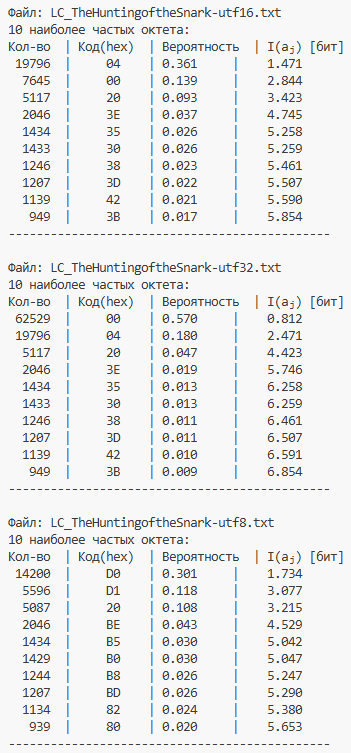
**4 наиболее частых октета, не являющихся кодами печатных символом ASCII:**

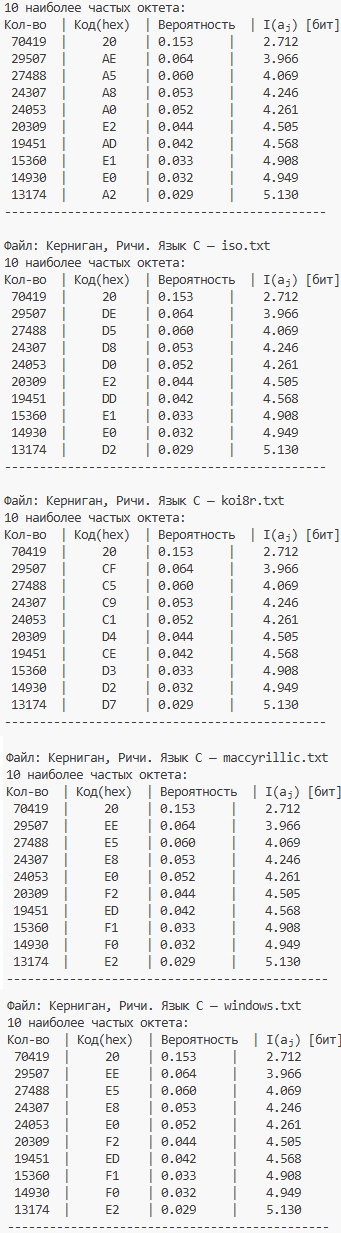
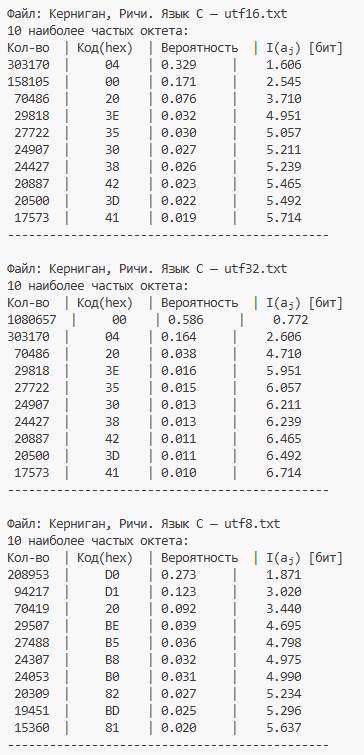
00 («нулевой символ») встречается в многобайтовых кодировках и является управляющим символом ASCII

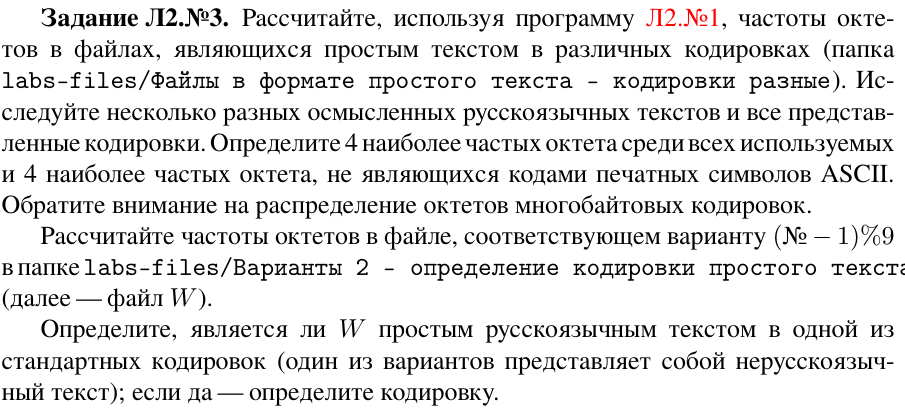
04 (значение зависит от соседних октетов) встречается в многобайтовых кодировках и является управляющим символм ASCII

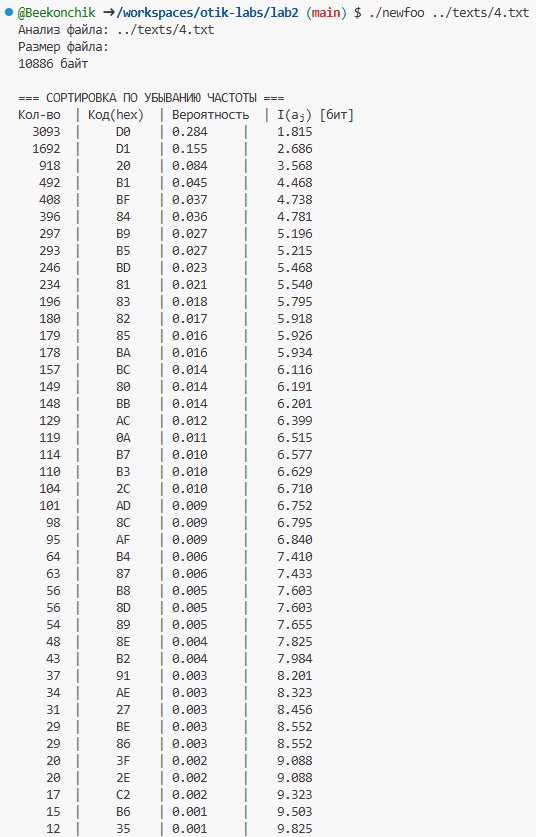
0A («перевод строки») встречается в многобайтовых кодировках и является управляющим символом ASCII

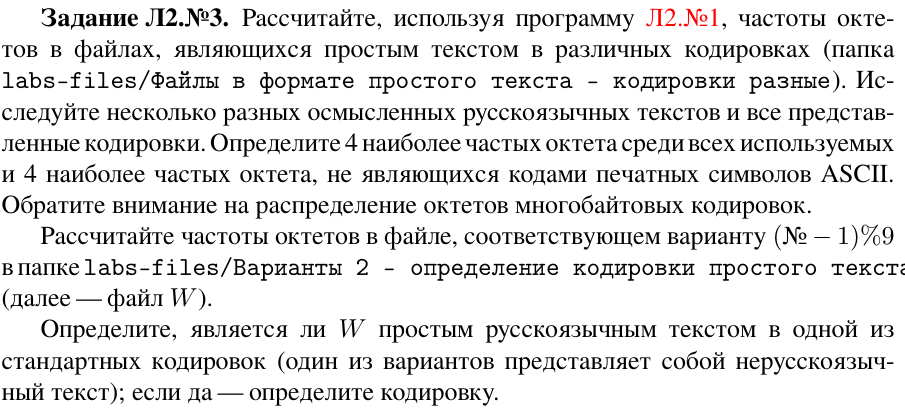
D0 (значение зависит от соседних октетов) встречается в многобайтовых кодировках и является управляющим символом ASCII

Многобайтовые кодировки: Однобайтовые кодировки:

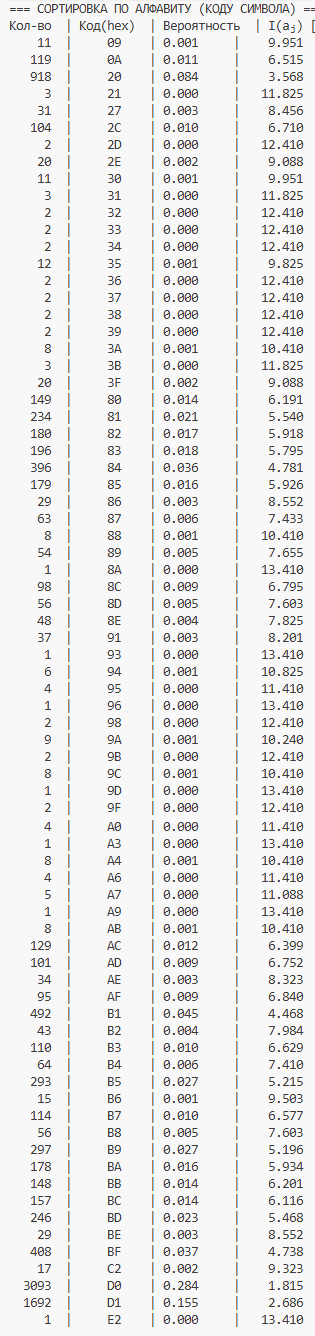
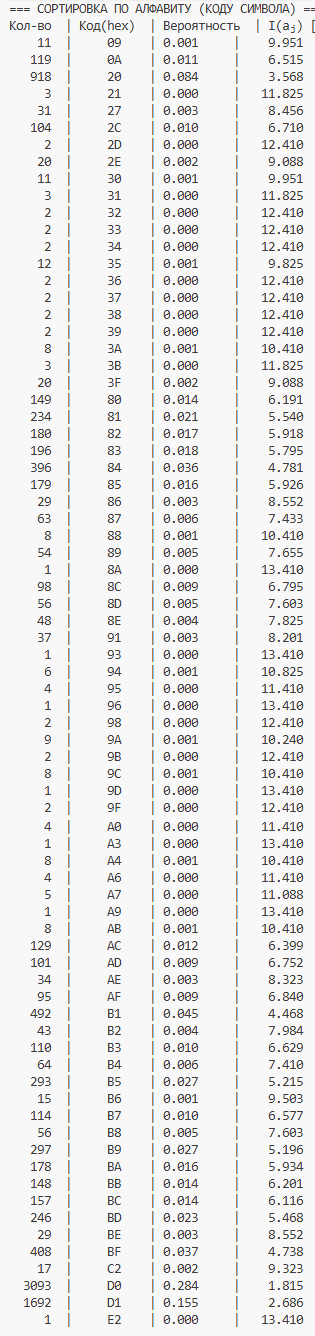
Многобайтовые кодировки: Однобайтовые кодировки:







Многобайтовая кодировка utf-8, так как наиболее часто встречающиеся октеты — D0 и D1, превосходящие даже «пробел» 20, что характерно только для utf-8.

Еще можно отметить гораздо меньшее количество октетов, соответствующих кодам символов ASCII, то есть букам латинского алфавита, то есть текст не является англоязычным.