Вариант 1

Написать программу, которая в качестве аргументов командной строки принимает заданное слово (первый аргумент) и имена текстовых файлов (документов). Необходимо для этого слова посчитать дисперсию его вхождений в документы:

$$\text{Дисперсия} = \sqrt{\frac{\sum_{i=1}^{N}(a_i - \bar{a})^2}{N-1}}$$

, где a_i - количество вхождений слова в i-ый документ,

N – общее количество документов,

$\bar{a}$ - среднее арифметическое вхождений слова по всем документам: $\bar{a} = \frac{\sum_{i=1}^{N} a_i}{N}$

Слова во входных файлах разделяются символами, для которых библиотечные функции isspace() или ispunct() возвращают ненулевое значение.

Вариант 2

Написать программу, которая в качестве аргументов командной строки принимает заданное слово (первый аргумент) и имена текстовых файлов (документов). Необходимо для этого слова посчитать медиану его вхождений в документы – т.е. надо *найти такое количество вхождений x заданного слова в документы, что количество документов, в которые слово входило не более x раз, максимально близко к количеству документов, в которые это слово входило более x раз.*

Слова во входных файлах разделяются символами, для которых библиотечные функции isspace() или ispunct() возвращают ненулевое значение.

Вариант 3

Написать программу, которая в качестве аргументов командной строки принимает имена текстовых файлов (первое - имя целевого документа, далее - имена остальных документов в коллекции), а на выходной поток выдаёт различные слова в целевом файле с указанием их частоты встречаемости в тексте целевого документа. Слова должны выводиться в порядке убывания их частоты встречаемости. При одинаковой частоте встречаемости выдавать слова в лексикографическом порядке. Частоту встречаемости считать по формуле TF-IDF:

$$w_i = \frac{tf_i \cdot idf_i}{\sqrt{\sum_j \left(tf_j \cdot idf_j\right)^2}},$$

где tf_i — частота встречаемости *i*-го слова в данном документе (term frequency),

$idf_i = \log \frac{N}{n_i}$ — логарифм отношения количества всех документов в коллекции к количеству документов, в которых встречается *i*-ое слово (inverse document frequency).

Слова во входных файлах разделяются символами, для которых библиотечные функции isspace() или ispunct() возвращают ненулевое значение. Считать, что словарь различных слов можно хранить в оперативной памяти.

Вариант 4

Написать программу, которая в качестве аргументов командной строки принимает заданное слово и имя текстового файла (документа). Документ состоит из предложений, разделителями которых являются точка, восклицательный знак и вопросительный знак. Необходимо для этого слова посчитать среднее арифметическое его вхождений в предложения документа:

$$\bar{a} = \frac{\sum_{i=1}^{N} a_i}{N}$$

, где a_i - количество вхождений слова в i-ое предложение,

N – общее количество документов.

Слова во входном файле разделяются символами, для которых библиотечные функции isspace() или ispunct() возвращают ненулевое значение.

Вариант 5

Написать программу, которая в качестве аргументов командной строки принимает заданное слово (первый параметр) и имя текстового файла (документа). Документ состоит из предложений, разделителями которых являются точка, восклицательный знак и вопросительный знак. Необходимо для этого слова посчитать дисперсию его вхождений в предложения документа:

$$\text{Дисперсия} = \sqrt{\frac{\sum_{i=1}^{N}(a_i - \bar{a})^2}{N-1}}$$

, где a_i - количество вхождений слова в i-ое предложение документа,

N – общее количество предложений в документе,

$\bar{a}$ - среднее арифметическое вхождений слова по всем предложениям документа: $\bar{a} = \frac{\sum_{i=1}^{N} a_i}{N}$

Слова во входном файле разделяются символами, для которых библиотечные функция isspace() или ispunct() возвращают ненулевое значение.

Вариант 6

Написать программу, которая в качестве аргументов командной строки принимает заданное слово и имя текстового файла (документа). Документ состоит из предложений, разделителями которых являются точка, восклицательный знак и вопросительный знак. Необходимо для этого слова посчитать медиану его вхождений в предложения документа – т.е. надо *найти такое количество вхождений x заданного слова в предложения, что количество предложений, в которые слово входило не более x раз, максимально близко к количеству предложений, в которые это слово входило больше x раз.*

Слова во входном файле разделяются символами, для которых библиотечные функции isspace() и ispunct() возвращают ненулевое значение.

Вариант 7

Написать программу, которая в качестве аргументов командной строки принимает заданное слово (первый параметр) и имя текстового файла (документа). Документ состоит из абзацев. Необходимо для этого слова посчитать среднее арифметическое его вхождений в абзацы документа:

$$\bar{a} = \frac{\sum_{i=1}^{N} a_i}{N}$$

, где a_i - количество вхождений слова в i-ый абзац,

N – общее количество абзацев в документе.

Слова во входном файле разделяются символами, для которых библиотечные функции isspace() или ispunct() возвращают ненулевое значение. Считать, что абзацы в тексте разделяются символом ‘\n’.

Вариант 8

Написать программу, которая в качестве аргументов командной строки принимает заданное слово и имя текстового файла (документа). Документ состоит из абзацев. Необходимо для этого слова посчитать дисперсию его вхождений в абзацы документа:

$$\text{Дисперсия} = \sqrt{\frac{\sum_{i=1}^{N}(a_i - \bar{a})^2}{N-1}}$$

, где a_i - количество вхождений слова в i-ый абзац документа,

N – общее количество абзацев в документе,

$\bar{a}$ - среднее арифметическое вхождений слова по всем абзацам документа: $\bar{a} = \frac{\sum_{i=1}^{N} a_i}{N}$

Слова во входном файле разделяются символами, для которых библиотечные функции isspace() или ispunct() возвращают ненулевое значение. Считать, что абзацы в тексте разделяются символом ‘\n’.

Вариант 9

Написать программу, которая в качестве аргументов командной строки принимает заданное слово и имя текстового файла (документа). Необходимо для этого слова посчитать медиану его вхождений в абзацы документа – т.е. надо *найти такое количество вхождений x заданного слова в абзацы документа, что количество абзацев, в которые слово входило не более x раз, максимально близко к количеству абзацев, в которые это слово входило более x раз.*

Слова во входном файле разделяются символами, для которых библиотечные функции isspace() или ispunct() возвращают ненулевое значение. Считать, что абзацы в тексте разделяются символом ‘\n’.

Вариант 10

Написать программу, которая в качестве аргумента командной строки принимает имя текстового файла (документа), а на выходной поток выдаёт различные слова в этом файле с указанием их частоты встречаемости в тексте документа. Слова должны выводиться в порядке убывания их частоты встречаемости. При одинаковой частоте встречаемости выдавать слова в лексикографическом порядке. Частоту встречаемости считать как отношение количества вхождений слова к общему количеству слов в тексте документа. Слова во входном файле разделяются символами, для которых библиотечные функции isspace() или ispunct()

возвращают ненулевое значение. Считать, что словарь различных слов можно хранить в оперативной памяти.

Вариант 11

Написать программу, которая в качестве аргументов командной строки принимает имена текстовых файлов (документов), а на выходной поток выдаёт различные слова в этих документах с указанием частоты встречаемости слов в документах. Слова должны выводиться в порядке убывания их частоты встречаемости. При одинаковой частоте встречаемости выдавать слова в лексикографическом порядке. Частоту встречаемости считать как отношение количества документов, в которых встречается данное слово, к общему количеству документов. Слова во входных файлах разделяются символами, для которых библиотечные функции isspace() или ispunct() возвращают ненулевое значение. Считать, что словарь различных слов можно хранить в оперативной памяти.

Вариант 12

Написать программу, которая в качестве аргументов командной строки принимает заданное слово (первый аргумент) и имена текстовых файлов (документов). Необходимо для этого слова посчитать среднее арифметическое его вхождений в документы:

$$\bar{a} = \frac{\sum_{i=1}^{N} a_i}{N}$$

, где a_i - количество вхождений слова в i-ый документ,

N – общее количество документов.

Слова во входных файлах разделяются символами, для которых библиотечные функции isspace() или ispunct() возвращают ненулевое значение.

Вариант 13

Написать программу, которая в качестве аргументов командной строки принимает целое число N и имя текстового файла (документа). Документ состоит из предложений, разделителями которых являются точка, восклицательный знак и вопросительный знак. Необходимо вывести все слова документа, которые встречаются более чем в N предложениях (с указанием количества вхождений каждого слова).

Слова во входном файле разделяются символами, для которых библиотечные функции isspace() или ispunct() возвращают ненулевое значение.

Вариант 14

Написать программу, которая в качестве аргументов командной строки принимает целое число N (первый аргумент) и имена текстовых файлов (документов). Необходимо вывести все слова, которые встречаются более чем в N документах (с указанием количества вхождений каждого слова).

Слова во входных файлах разделяются символами, для которых библиотечные функции isspace() или ispunct() возвращают ненулевое значение.

Вариант 15

Написать программу, которая в качестве аргументов командной строки принимает целое число N и имя текстового файла (документа). Документ состоит из абзацев. Необходимо вывести все слова, которые встречаются более чем в N абзацах (с указанием количества вхождений каждого слова).
Слова во входном файле разделяются символами, для которых библиотечные функции isspace() или ispunct() возвращают ненулевое значение. Считать, что абзацы в тексте разделяются символом ‘\n’.

Вариант 16

Написать программу, которая в качестве аргумента командной строки принимает имя текстового файла, содержащего целые числа. Длина имени не превышает значение константы PATH_MAX (см. файл <limits.h>). Необходимо написать программу, которая модифицирует заданный файл, сортируя числа в файле по неубыванию. Использовать дополнительные файлы не разрешается.

Вариант 17

Написать программу, которая в качестве аргумента командной строки принимает имя текстового файла, содержащего целые числа. Длина имени не превышает значение константы PATH_MAX (см. файл <limits.h>). Модифицируйте заданный файл так, чтобы в нем сначала шли нули, затем отрицательные числа, затем положительные числа. Порядок следования внутри файла отрицательных и положительных чисел может быть произвольным.