

Задача 6. Разделить на слова

Источник:	основная*
Имя входного файла:	input.txt
Имя выходного файла:	output.txt
Ограничение по времени:	1 секунда
Ограничение по памяти:	разумное

Требуется реализовать функции для выделения слов из заданной строки. Слова отделены друг от друга символами-разделителями, которые передаются в строку как параметр.

Сигнатура функций должна быть такой:

```
typedef struct Tokens_s {
    int num;           //количество слов в строке
    char **arr;        //массив слов (каждый элемент -- строка, т.е. char*)
} Tokens;
//tokens: структура, в которую нужно положить результаты
//str: строка, в которой нужно искать слова
//delims: все символы-разделители в виде строки
void tokensSplit(Tokens *tokens, const char *str, const char *delims);
//удаляет все ресурсы внутри tokens
void tokensFree(Tokens *tokens);
```

Память для массива слов `tokens->arr` следует выделять динамически ровно на столько элементов, сколько слов в строке. Поскольку заранее это количество неизвестно, то нужно запускать алгоритм два раза: первый раз, чтобы узнать сколько слов, и второй раз, чтобы записать слова в массив.

Ваша реализация вышеописанных функций должна работать согласно следующим договорённостям:

1. Вызывающий гарантирует, что параметр `tokens` не нулевой (для обеих функций) и указывает на структуру `Tokens`.
2. Если при вызове `tokensSplit` указатель `tokens->arr` нулевой, то внутри функции нужно только посчитать количество слов и записать его в `tokens->num`.
3. Если при вызове `tokensSplit` указатель `tokens->arr` не нулевой, то он должен указывать на массив, в который точно войдут все слова. В этом случае реализация функции должна записать в `tokens->num` количество слов, а в `tokens->arr[i]` записать *i*-ое слово, самостоятельно выделив под него память с помощью `malloc`.
4. Функция `tokensFree` должна удалять массив слов и сами строки-слова с помощью `free`. При этом программа должна работать корректно, даже если эту функцию случайно вызовут два или три раза подряд.

Таким образом, вызывающий может завести структуру `tokens`, потом определить количество слов первым вызовом `tokensSplit`, затем выделить память на массив `tokens->arr`, и, наконец, найти все слова вторым вызовом `tokensSplit`.

С помощью этих функций нужно решить тестовую задачу. Дана одна строка длиной до 10^6 , состоящая из букв латинского алфавита и знаков препинания четырёх типов: точка, запятая, точка с запятой, двоеточие. Нужно найти слова в этой строке, состоящие из букв, и вывести их в файл в таком же формате, как показано в примере.

Пример

input.txt	output.txt
..ko,.Privet:kreved,.,;ko:;,.	4 ko Privet kreved ko

Комментарий

Следует выводить слова ровно в том порядке, в котором они встречаются в строке.