

# Задание 1: Считаем слова


## Что такое `wc`?

`wc` (word counter) – это утилита, используемая для подсчёта *символов*, *слов* и *строк* в произвольном тексте. Эта утилита входит в базовый набор [GNU Coreutils](#), что означает, что вы скорее всего найдёте её уже установленной на *Linux*, *MacOS*, *\*BSD* и даже *Windows*, если вы установили окружение MinGW для работы с C/C++. Можете проверить доступность у себя `wc`, открыв терминал и написав там `wc`.

### Использование `wc`

Если мы отдадим `wc` в качестве аргумента файл, то в ответ получим 3 числа и ещё раз название файла. Файл `the_laughing_heart.txt` поставляется вместе с заданием.

```
$ wc the_laughing_heart.txt
  20      98    495 the_laughing_heart.txt
```

 Shell

В выводе `wc`:

- 20 – число строк, а точнее число символов переноса строк `'\n'`;
- 98 – число слов;
- 495 – число символов, включая пробелы и символы переноса строк.

## Постановка задачи

### Часть 1: чтение из файла

Необходимо реализовать утилиту `wc`, которая будет считать количество символов, слов и строк внутри заданного файла. Но в отличие от оригинального `wc`, наша реализация должна выводить 3 строки: на первой число строк, на второй – число слов, а на третьей – число символов. Название файла выводить не нужно. Если файл с таким названием не существует, будем выводить ошибку.

Будем считать, что все символы входного файла взяты из таблицы ASCII (то есть не надо учитывать кириллические символы). В качестве пробельных символов наша реализация `wc` должна ожидать:

- обычный пробел: `' '`;
- табуляция: `'\t'`;
- перенос строки: `'\n'`;
- возврат каретки (перемещает курсор в начало строки без перехода на следующую строку): `'\r'`;
- вертикальная табуляция: `'\v'`.

### Пример использования нашего решения

Допустим, мы написали решение в файле `main.c`. Скомпилируем это решение с помощью `gcc`, указав название для исполняемого файла с помощью флага `-o`:

```
$ gcc -o my-wc main.c
```

 Shell

Ожидается, что наш исполняемый файл можно использовать следующим образом:

```
$ ./my-wc the_laughing_heart.txt
20
98
495
```

 Shell

### Подсказка

Для решения могут быть полезны функции `fopen()`, `fclose()` и `fgetc()`. Но вы можете написать и иное решение, которое ими не пользуется.

## Часть 2: чтение из стандартного потока ввода

### Использование `wc`

Оригинальный `wc` ещё умеет читать текст из стандартного потока ввода. На примере ниже, мы запустили `wc`, не передав файл в аргументах, и он начал ждать от нас ввода с клавиатуры. Мы ввели строку текста, нажали на Enter и всё ещё ничего не произошло – тут перенос строки не является обозначением конца ввода. Чтобы сообщить `wc`, что это конец и больше мы не планируем что-либо печатать, нужно нажать сочетание клавиш `Ctrl+D`, после чего мы получим вывод и работа `wc` завершится.

```
$ wc
i am typing this text myself
1      6      29
```

 Shell

Однако в стандартный поток ввода мы можем вводить текст не только с клавиатуры, мы можем воспользоваться, например, программой `cat` для получения содержимого файла и перенаправить её вывод в стандартный поток ввода `wc` с помощью вертикальной черты:

```
$ cat the_laughing_heart.txt | wc
20      98     495
```

 Shell

Давайте теперь изменим работу нашей программы так, чтобы она читала текст не из файла, а из стандартного потока ввода (`stdin`), если в качестве аргумента не был передан файл. Точно так же, как и `wc`.

Соответственно:

- если в качестве первого аргумента командной строки указано название файла, то необходимые данные считаются для этого файла;
- если название файла не указано, то вместо него данные считаются для строк, которые подаются в стандартный поток ввода.

### Пример использования нашего решения

Опять же, ожидается, что нашим решением можно пользоваться точно так же, как и оригинальным `wc`, за исключением формата вывода:

```
$ cat the_laughing_heart.txt | ./my-wc
20
98
495
```

 Shell

### Подсказка

Стандартный поток ввода можно читать как обычный файл. В заголовочном файле `stdio.h` уже объявлено имя `stdin`.