# Problem Set 7: Files

## Goals

1. Learn to work with command line arguments.
2. Acquire work with files.
3. Deal with specific limitations (when a simple problem is not that simple).

## Task 1: Count the words

Create a program `count_words.c`, which counts how many times there is a in the input file **sequence of characters : pineapple** . The result is written to the same file (the original contents of the file will be overwritten).

> **Note :** Throughout the program, case when counting and comparing words **does not matter** .

It is forbidden to use any library functions in the program ( `strlen()`, `strcpy()`, `strcmp()`etc.). Only the following 4 functions are allowed:

- `fopen()`to initialize work with the file
- `fclose()`to finish working with the file
- `fgetc()`to read from a file
- `fputc()`to write to a file

The program will have one argument:

- input-output file (its name)

The following is an example of using the program. Bold text will represent user input. The line beginning with the **'$' character** will represent the command line.

```
$ ls
bananas.txt count_words
$ ./count_words bananas.txt
$ ls
bananas.txt count_words
```

If the contents of the file `bananas1.txt`next:

```
Bananas are edible fruits, botanically berries. In some countries, bAnAnAs used for cooking are called plantains, distinguishing them
```

file content `bananas1.txt`after successful completion of the program is as follows:

```
5
```

If the contents of the file `bananas2.txt`next:

```
In the past, there was a hoax that some countries wanted to ban ananas (also known as pineapple) because it was less delicious than b
```

file content `bananas2.txt`after successful completion of the program is as follows:

```
2
```

# Task 2: Every other word

Create a program `every_second.c`, which creates a file based on another file by writing to the second file every second word read from the first file between the words **START** and **STOP** . The names of these files are specified as command line arguments. Any sequence of characters separated by spaces is considered a word.

Word storage (every other word) starts after the word appears in the input file **START** . Words (every other word) are stored in the output file, and word storage ends when the word found in the input file **STOP is** .

> **Note :** You must separate the words stored in the output file with a space.

> **Note :** Throughout the program, while saving every other word **depends** on the size of the letters.

It is forbidden to use any library functions in the program ( `strlen()`, `strcpy()`, `strcmp()`etc.). Only the following 4 functions are allowed:

- `fopen()`to initialize work with the file
- `fclose()`to finish working with the file
- `fgetc()`to read from a file
- `fputc()`to write to a file

The program will have two arguments:

- the first argument will be the input file (its name)
- the second argument will be the output file (its name)

The following is an example of starting and using the program. Bold text will represent user input. The line beginning with the **'$' character** will represent the command line.

```
$  ls
every_second input.txt
$  ./every_second input.txt output.txt
$  ls
every_second input.txt output.txt
```

After the successful completion of the program, there is a newly created file in the directory. If the contents of the file `input.txt`next:

```
Hello start START Hello Kitty, say, how are you doing today? STOP
```

file content `output.txt`after successful completion of the program is as follows:

```
Kitty, how you today?
```