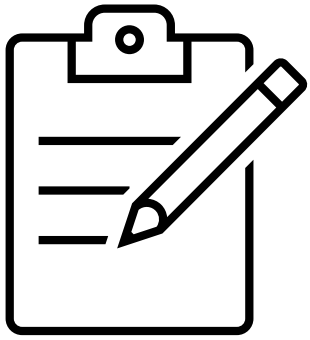
Tutorial 4– Analysis of a text



What we learn in this tutorial:

- Structures in C language
- Use of Files (*fopen*() and *fscanf*())
- Sorting algorithm

What is the most used letter in English? Is it the same in other languages that use Latin characters? Has is it always been the same throughout history?

Calculating the frequency of presentation of letters in the alphabet provides important information about a language and can be used to derive efficient text compression algorithms (eg. Huffman coding).

```
e = 12.89%
t =  8.94%
a =  7.62%
o =  7.55%
h =  6.70%
i =  6.65%
s =  6.48%
n =  6.38%
r =  5.84%
d =  4.45%
l =  4.38%
u =  3.13%
c =  2.61%
m =  2.55%
w =  2.38%
y =  2.33%
g =  2.03%
f =  1.81%
p =  1.60%
b =  1.56%
v =  1.03%
k =  0.75%
x =  0.12%
q =  0.08%
j =  0.08%
z =  0.06%
```

We want to build a program that estimates the frequency of presentation of each of the 26 letters in the English alphabet by analysing a large text file. The program opens the text file and counts the how many times each letter appears disregarding the case (i.e. lower or uppercase letter are considered the character). The percentage is then obtained dividing these numbers by the total numbers of letters.

The output on the left was obtained by applying the program to a modern novel.

In this tutorial we use a file with the entire Shakespeare's work (*shakespeare.txt*) in text format. The file must be copied in the same directory of the source code and executable so to be found by the program.

1) Write a program that:

1. Declare a file handler (FILE *fp) and open the file for reading with the function *fopen()* from stdio.h.
2. Declare a structure (*struct letter*) that contains a *char ch* and an *int qnt* that counts the number of occurrences of a character.. Declare an array of 26 *struct letter*.
3. Initialise the array by assigning each of the characters in the English alphabet (from 'a' to 'z') to the fields *ch* and zero to the *qnt* fields.
4. Using a while loop read all the characters in the file until EOF (using *fscanf()*). If the character is between 'a' and 'z' or 'A' and 'Z' increase the corresponding *qnt* in the array.
5. Once that the letters have been counted, use the *bubblesort* algorithm to sort the array of structures in decreasing order.
6. Print the percentage of each letter using two decimal digits and proper alignment.