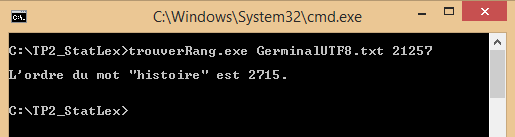
TP2

Statistiques lexicales

## Objectif

Vous devez créer un programme qui permet de fouiller un fichier texte, par exemple, un livre, et de déterminer le rang d’un mot donné après avoir été trié alphabétiquement.



L’exemple précédent montre le fonctionnement obligatoire de votre programme :

* L’utilisateur entre le nom de l’exécutable, suivi du chemin du texte à parcourir et de l’ordre d’apparition du mot dans le texte.
* Le programme répond en identifiant le mot et en donnant son rang alphabétique parmi les autres mots du texte.
* C’est tout, mais ce n’est pas si simple.

## Contraintes

* Vous devez utiliser le fichier source qui vous a été fourni.
  + Vous ne pouvez pas inclure d’autres librairies que celles qui sont déjà inclues.
  + Ne modifiez ni les constantes MAXMOTS et MAXCARMOT, ni les prototypes de fonctions.
  + N’utilisez pas de variables globales.
* Les prototypes extraireMots() et trierMots() sont les deux fonctions principales.
  + extraireMots() lit le texte et sépare les mots (parsage), les insérant dans un tableau qui contient MAXMOTS mots au maximum.
    - Les mots doivent avoir une taille maximale de MAXCARMOT.
      * Les mots plus longs ne seront pas ignorés, simplement tronqués.
        + \0…
      * S’il y a plus MAXMOTS mots, vous devez les ignorer.
    - Tous les mots devront être conservés en casse minuscule.
    - Pour des fins de simplicité, on ne considère que les caractères alphanumériques.
      * C’est-à-dire qu’un mot commence avec un caractère alphanumérique (A-Z, a-z, 0-9) et se termine lorsqu’un caractère non-alphanumérique est rencontré.
      * Donc, pas d’espaces, de tirets, de caractères accentués, d’apostrophes, etc.
        + Par exemple, la phrase suivante :

L’épistémologie du savoir d’aujourd’hui.

* + - * + devra donner les mots suivants :

l

pist

mologie

du

savoir

d

aujourd

hui

* + trierMots() triera les mots en ordre croissant alphabétique en utilisant strcmp() pour déterminer quel mot est « plus petit » qu’un autre.
    - Au final, il ne doit y avoir aucun doublon dans cette liste triée.
* Vous ne pouvez utiliser que les fonctions de librairie suivantes :
  + printf()
  + fopen()
  + fclose()
  + fgetc()
  + strlen()
  + strcpy()
  + strcmp()
  + atoi()
* La correction se fera à partir d’une fenêtre de console (cmd).
  + Le chemin est fourni comme premier argument de ligne de commande.
  + L’index du mot dans le texte original sera le deuxième argument.
    - Vous devez contrôler les erreurs des utilisateurs.
  + La vitesse d’exécution de votre programme sera un facteur important lors de son évaluation.

## Lecture dans un fichier

La lecture dans un fichier DOIT s’effectuer de la manière suivante :

|  |
| --- |
| int extraireMots(char cheminFichier[], char mots[MAXMOTS][MAXCARMOT])  {  …  FILE \*f;  //Ouvrir fichier  f = fopen(cheminFichier, "r");  …  //Traitement  …  //fermer fichier  fclose(f);  } |

fopen() ouvre un fichier, fclose() le ferme. cheminFichier, c’est la chaîne de caractères qui contient le path vers le fichier.

Durant le « traitement », prenez soin de déclarer un int qui recevra la valeur de fgetc(). Sinon, certaines valeurs pourraient ne pas être retenues. Il n’y a pas de problème lorsqu’un char reçoit la valeur d’un caractère VALIDE contenue dans un int.

|  |
| --- |
| //Traitement, peut être une fonction séparée  …  int c;  …  c = fgetc(f);  … |

Note : Le nom des variables peut évidemment être différent.

## Remise

Vous devez remettre le fichier contenant uniquement le code source de votre programme. Vous pouvez remettre le projet à tout moment d’ici le 17 octobre, avant la fin de la journée.