# SÉANCE 8

## Objectif

Le but de cette huitième séance est de manipuler les fichiers de texte.

## Outils

Le fichier `lireligne.c` contient la définition de la fonction `LireLigne` que vous utiliserez dans chacun des exercices ci-dessous. Son fonctionnement est décrit par des commentaires.

Vous utiliserez également les fonctions de la bibliothèque standard vues en cours : `fopen`, `fclose`, `fscanf`, `fprintf`, `perror`, `fgetc`, `fputc` et `feof`.

Dans tous les exercices, on suppose qu'un nom de fichier contient au plus 255 caractères. On suppose également que, pour les fonctions qui reçoivent en paramètres des identificateurs de fichier (de type `FILE *`), la fonction appelante a bien effectué l'ouverture des fichiers correspondants (en lecture ou en écriture, selon le cas).

## Exercices

### Exercice 1 (Statistiques sur les mots d'un fichier)

Le but de cet exercice est d'écrire un programme permettant de compter et de calculer la longueur moyenne des mots présents dans le contenu d'un fichier.

1. Écrivez la définition de la structure `sStats` constituée de deux champs : `NbMots`, de type `int`, qui permettra de stocker le nombre de mots du texte, et `LongMoy`, de type `float`, qui permettra de stocker la longueur moyenne (en nombre de caractères) d'un mot de ce texte.
2. Comme cela a été présenté en cours, en utilisant l'instruction `typedef` permettant la définition de synonymes de types, définissez le synonyme `tStats` de cette structure.
3. Écrivez la fonction d'en-tête :
   `void Statistiques(FILE *f, tStats *pStats)`
   qui calcule le nombre et la longueur moyenne des mots contenus dans un fichier d'identificateur `f` et stocke ces résultats dans la structure pointée par `pStats`. Chaque mot sera lu avec la fonction `fscanf` et le spécificateur de format `%s`. On suppose qu'un mot contient au plus 63 caractères. La fin du fichier sera détectée grâce à la fonction `feof`.
4. Écrivez la fonction principale afin que le programme :
   - demande à l'utilisateur le nom du fichier à traiter : cette lecture doit être faite avec la fonction `LireLigne` ;
   - ouvre le fichier de texte en lecture ;
   - calcule les statistiques sur les mots qu'il contient ;
   - ferme le fichier ;
   - affiche les statistiques.

   Vous pouvez tester votre programme avec le fichier `texte.txt` qui contient 28 mots dont la longueur moyenne est approximativement de 4.5 caractères.
5. Quand votre programme fonctionne, modifiez la fonction principale afin que, si l'ouverture du fichier échoue (si le fichier n'existe pas, par exemple), alors le programme doit :
   - afficher le nom du fichier et un éventuel message sur la sortie standard des erreurs avec la fonction `perror` ;

- se terminer en retournant le code 1.

Même si ce n'est pas précisé, vous procéderez de la même manière dans les exercices suivants pour la lecture des noms de fichiers et la vérification du succès de leur ouverture.

### ☞ Exercice 2 (Remplacement de caractère)

Le but de cet exercice est d'écrire un programme permettant de lire le contenu d'un fichier et de le recopier dans un autre fichier en remplaçant toutes les occurrences d'un caractère par un autre.

1. Écrivez la fonction d'en-tête :
   `void Remplacer(char Ch[], FILE *Source, FILE *Destination)`
   qui recopie le contenu du fichier d'identificateur `Source` dans le fichier d'identificateur `Destination` en remplaçant toutes les occurrences de `Ch[0]` par `Ch[1]`. La lecture doit être faite caractère par caractère avec la fonction `fgetc`. La fin du fichier doit être détectée par la valeur de retour de `fgetc` qui devient `EOF` en fin de fichier (utilisez un `int` pour récupérer le caractère lu). Utilisez la fonction `fputc` pour l'écriture de chaque caractère.
2. Écrivez la fonction principale afin que le programme :
   - demande à l'utilisateur la chaîne de deux caractères décrivant le remplacement : le premier est le caractère à remplacer et le second est le remplaçant ;
   - demande à l'utilisateur le nom du fichier source ;
   - ouvre le fichier en lecture ;
   - demande à l'utilisateur le nom du fichier destination ;
   - ouvre le fichier en écriture de telle sorte que, s'il existe déjà, son contenu est écrasé ;
   - effectue la recopie avec remplacement ;
   - ferme les fichiers.

## Pour aller plus loin

### ☞ Exercice 3 (Numérotation des lignes)

Le but de cet exercice est d'écrire un programme permettant de lire le contenu d'un fichier et de le recopier dans un autre fichier en faisant précéder chaque ligne par son numéro suivi d'un espace.

1. Écrivez la fonction d'en-tête :
   `void Numeroter(FILE *Source, FILE *Destination)`
   qui recopie le contenu du fichier d'identificateur `Source` dans le fichier d'identificateur `Destination` en faisant précéder chaque ligne de son numéro, la première ligne portant le numéro 1. On suppose qu'une ligne contient au plus 255 caractères. On suppose que chaque ligne du fichier se termine par un caractère `\n` (cela permet d'éviter des problèmes avec la dernière ligne). Utilisez la fonction `LireLigne` pour lire le fichier ligne par ligne. Détectez la fin du fichier avec la valeur de retour de la fonction `LireLigne`. Écrivez chaque ligne avec la fonction `fprintf` et le format `%5d` pour écrire le numéro de ligne aligné à droite sur 5 caractères.
2. Écrivez la fonction principale afin que le programme :
   - demande à l'utilisateur le nom du fichier source ;
   - ouvre le fichier en lecture ;
   - demande à l'utilisateur le nom du fichier destination ;
   - ouvre le fichier en écriture de telle sorte que, s'il existe déjà, son contenu est écrasé ;
   - effectue la recopie avec numérotation des lignes ;
   - ferme les fichiers.