Programmation Procédurale : TD Gestion de fichiers

Ne pas oublier de créer un répertoire pour placer les codes de ce TP !

Toutes les fonctions écrites doivent être testées dans le programme principal !

Exercice 1 (*Test des fonctions de base*)

1. Creer un nouveau fichier `texte.txt` dans votre répertoire courant. L'ouvrir et écrire "Je suis ton père" dedans.
2. Dans un programme C ouvrir le fichier en mode lecture à l'aide de la fonction `fopen`.
3. Récupérer et afficher le premier caractère du fichier ouvert grace à la fonction `fgetc`.
4. Récupérer et afficher la phrase entière écrite dans le fichier ouvert grace à la fonction `fgets`.
5. Écrire le mot "Luke" en début de fichier grace à la fonction `fputs`
6. Creer un second fichier `chiffre.txt` dans votre repertoire courant. L'ouvrir et écrire "42 7 2.5" dedans.
7. Sommez le premier et dernier nombre du fichier à l'aide de la fonction `fscanf`.

Exercice 2 (*Etude statistique d'un texte partie 1*)

1. Écrire un code qui permet d'écrire dans le terminal l'ensemble du fichier `HarryPotter.txt`.
2. Compter et afficher le nombre de caractères du fichier.
3. Écrire une fonction `int nmbCaractere(FILE * fichier, char a)` qui retourne le nombre de fois qu'un caractère `a` est dans le fichier. Tester la fonction avec les lettres `"h"` et `"H"`.
4. Remplir un tableau qui va contenir le nombre d'occurences de chacun des caractères du fichier.
5. Modifier le tableau précédent pour qu'il contienne les probabilités d'apparition de chaque caractère.

Exercice 3 (*Base de donnée d'Etudiant*)

1. Recupérer le code des exercices du TD sur les structures (construction d'une structure `Etudiant` et remplissage d'un tableau d'`Etudiant`).
2. Écrire une procédure `ecrireEtudiant(Etudiant tabEtu[])` qui prends en paramètre un tableau d'`etudiant` et remplit un fichier texte de la liste des `Etudiant` du tableau. Vous écrirez le nom, le prénom et les autres champs d'un `Etudiant` sur chaque ligne du fichier.
3. Modifier le fichier texte en rajoutant manuellement quelques Etudiants.
4. Écrire une fonction `Etudiant * lireEtudiant()` qui lit un fichier texte contenant une liste d'`Etudiant` et les stocker dans un tableau qui sera retourné. Modifier le code pour que le tableau d'`Etudiant` sur lequel on souhaite travailler soit récuperé depuis le fichier créé précédemment, plutôt que saisi manuellement.
5. Modifier la fonction `ecrireEtudiant(Etudiant tabEtu[])` pour ajouter, à la fin de la ligne donnant les informations d'un étudiant, sa moyenne.
6. Écrire des version des fonctions `ecrireEtudiant(Etudiant tabEtu[])` et `Etudiant *tab lireEtudiant()` pour des fichiers binaires.

Exercice 4 (*Ecriture d'un fichier JSON*)

Il existe un type de fichier qui permet de stocker des structures de valeurs sous forme de chaine de caractères : le format `JSON`

Le but de cet exercice est de créer une structure avec diverses informations (nous reprendrons la structure `Etudiant` des TPs précédents) pour la parcourir et la stocker dans un fichier `ASCII` au format `JSON`.

Si vous souhatez en savoir plus sur ce format de fichier, vous pouvez consulter la page wikipédia disponible ici : `https://fr.wikipedia.org/wiki/JavaScript_Object_Notation`

Soit la structure Etudiant au format suivant :

```
typedef struct {
    char*  nom;
    char*  prenom;
    int    groupe;
    float* notes;
}
```

Si on dispose d'un tableau de 3 étudiants, la chaîne de caractères au format JSON sera la suivante (les retours à la ligne ou les tabulations sont optionnelles pour le format JSON, mais sont utilisées ici pour plus de lisibilité) :

```
[
    {
        "nom"    : "......",
        "prenom" : "......",
        "groupe" : 1,
        "notes"  : [
                      12.57,
                      16.58,
                      13.2,
                      5.7,
                      8.8,
                      19.5
                   ]
    },
    {
        "nom"    : "......",
        "prenom" : "......",
        "groupe" : 2,
        "notes"  : [
                      2.57,
                      6.58,
                      3.2,
                      15.7,
                      18.8,
                      9.5
                   ]
    },
    {
        "nom"    : "......",
        "prenom" : "......",
        "groupe" : 4,
        "notes"  : [
                      11.1,
                      12.2,
                      13.3,
                      14.4,
                      15.0,
                      20.0
                   ]
    }
]
```