TP 4 : Entrées/sorties, fichiers, chaînes de caractères

Exercice 1 : Lecture et écriture de caractères dans les E/S standards

Écrire un programme qui lit un caractère dans le fichier d'entrée standard et l'écrit dans le fichier de sortie standard tant que le caractère lu est différent du caractère "nouvelle ligne".

Exercice 2 : Lecture et écriture caractère par caractère dans un fichier

Écrire un programme qui crée le fichier `mes_caracteres.txt`, y écrit un par un les caractères saisis par l'utilisateur jusqu'à la frappe de la touche « Entrée » et ferme ce fichier. Il rouvre ensuite ce fichier en lecture, lit un par un les caractères qui y sont enregistrés et les affiche.

Exercice 3 : Lecture et écriture caractère par caractère dans un fichier

Écrire un programme qui lit le contenu du fichier texte `entree.txt`, et le recopie caractère par caractère dans le fichier texte `sortie.txt`.

Exercice 4 : Écriture et lecture d'un fichier ligne par ligne

Ecrire un programme qui crée le fichier `mon_texte.txt`, y écrit une suite de lignes, puis le ferme. Il rouvre ensuite ce fichier en lecture, lit les lignes qu'il contientet les affiches. La suite de lignes lue peut contenir au maximum `NBLIGNES` ayant au maximum `NBCAR` caractères (un caractère étant réservé pour le caractère de fin de chaîne).

Exercice 5 : Lecture et écriture avec format dans un un fichier

- Ecrire un programme utilise la fonction `fscanf` pour stocker dans des variables les informations d'un fichier annuaire qui comporte sur chaque ligne un nom, un prénom, le numéro de téléphone et l'adresse mail d'une personne. on suppose que le nom de la personne ne comporte pas d'espace.
  exemple d'annuaire
  `annuaire.txt`
  `Chamroukhi Faicel 04 94 14 20 06 chamroukhi@univ-tln.fr`
  `nom prenom 04 94 14 00 00 nom@univ-tln.fr`
  affichage
  `nom : Chamroukhi`
  `prenom : Faicel`
  `tel :04 94 14 20 06`
  `mail : chamroukhi@univ-tln.fr`
- Ecrire un programme qui ajoute une entrée dans l'annuaire (en utilisant la focntion `fprintf`). On utilisera la fonction `atoi` de la bibliothèque standard pour convertir une chaîne de caractères en un entier et la fonction `strcpy` pour la copie d'une chaîne de caractères dans une autre.
  exemple d'exécution :
  Utilisation entrée : `annuaire nom_fichier nom prenom tel mail`
  `gcc ecriture_annuaire.c -o ecriture_annuaire`
  `./ecriture_annuaire`

  `./ecrit annuaire.txt Nom1 Prenom1 04 94 14 20 01 nom1@univ-tln.fr`
  Les informations sont stockees dans l'annuaire `annuaire.txt`
  `Chamroukhi Faicel 04 94 14 20 06 chamroukhi@univ-tln.fr`
  `nom prenom 04 94 14 00 00 nom@univ-tln.fr`
  `Nom1 Prenom1 04 94 14 20 01 nom1@univ-tln.fr`

Exercice 5 : Fichiers et chaines de caractères

Écriture d'un programme qui crée et interroge une base de données qui décrit les livres d'une bibliothèque et leurs auteurs. Un livre est décrit par sa cote (un numéro qui l'identifie dans la bibliothèque), son titre, son année de publication (un entier positif) et ses auteurs. On décide de stocker ces livres dans deux fichiers :

- le fichier `livres.txt` qui contient une suite de triplets *cta* (un élément de doublet par ligne) indiquant que le livre de cote c a pour titre t et a été publié l'année a. A

chaque livre de la bibliothèque, il correspondra un et un seul triplet dans ce fichier.
- le fichier `auteurs.txt` qui contient une suite de doublets nc (un élément de doublet par ligne) indiquant que l'auteur de nom n a écrit le livre de cote c. A chaque livre de la bibliothèque de cote c écrit par $k(k \geq 1)$ auteurs de noms $n_1, ..., n_k$, il correspondra k et seulement k doublets dans ce fichier : $n_1c, ..., n_kc$. Supposons par exemple, que la bibliothèque contienne les deux livres :

1. *Initiation à l'informatique* de cote L1, écrit par Pierre Legrand et publié en 1995 ;
2. *Le langage C mais c'est très simple*, de cote L2 écrit par Anne Dupont et Pierre Legrand et publié en 2004.

Le fichier `livres.txt` aura le contenu suivant :

```
L1
Initiation a l'informatique (les accents sont omis)
1995
L2
Le langage C, mais c'est tres simple
2004
```

et le fichier `auteurs.txt` aura le contenu suivant :

```
Pierre Legrand
L1
Claire Dupont
L2
Pierre Legrand
L2
```

Les identificateurs des fichiers `livres.txt` et `auteurs.txt` pourraient être affectés aux variables globales `flivres` et `fauteurs`.
Les constantes `LCOTE`, `LTITRE`, `LANNEE` et `LNOM` auront pour valeurs respectives le nombre maximum plus un de caractères d'une cote de livre, d'un titre de livre, d'une année et d'un nom d'auteur (il faut compter le nombre caractères de fin de chaîne)..

1. Définir une fonction d'en-tête

   `void ecrire_livre(char *cote, char *titre, int annee)`

   qui a pour effet d'écrire à la fin du fichier `flivres` un triplet *cote titre année* (une donnée par ligne). Il faudra vérifier au préalable que les chaînes de caractères cote et titre ne dépassent les longueurs maximum fixées (`LCOTE` et `LTITRE`) et qu'annee est un entier positif. Si une erreur d'écriture est détectée, un message d'erreur sera affiché et l'exécution du programme sera interrompue.

2. Définir une fonction d'en-tête `int ecrire_auteur(char *nom, char *cote)` qui a pour effet d'écrire à la fin du fichier `fauteurs` un doublet *nom cote* (un élément par ligne). Il faudra vérifier au préalable que les chaînes de caractères *nom* et *cote* ne dépassent les longueurs maximum fixées (`LNOM` et `LCOTE`). Si une erreur d'écriture est détectée, un message d'erreur sera affiché et l'exécution du programme sera interrompue.
3. Définir une fonction `int lire_livre(char *cote, char *titre, int *annee)` qui a pour effet de lire dans le fichier `flivres` le prochain triplet *cote titre année*, et qui retourne 0 si la fin du fichier a été atteinte ou si une erreur s'est produite, ou 1 sinon. Cette lecture sera réalisée par la fonction `lire_ligne` (voir l'annexe) et l'année convertie en nombre par la fonction `atoi` de la bibliothèque standard.
4. Définir une fonction d'en-tête `int lire_auteur(char *nom, char *cote)` qui a pour effet de lire dans le fichier fauteurs, le prochain doublet *nom cote* et qui retourne 0 si la fin du fichier a été atteinte ou si une erreur s'est produite, ou 1 sinon. La lecture des chaînes de caractères nom et cote pourra être effectuée par la fonction `lire_ligne`.
5. Définir une fonction d'en-tête `void livres_publies_apres(int annee_inferieure)` qui a pour effet d'afficher, à raison de un par ligne, les titres des livres publiés après l'année `annee_inferieure`. Cette fonction fera appel à la fonction `lire_livre`. Par exemple, l'appel `livres_publies_apres(2000)` aura pour effet d'afficher :

   ```
   Le langage C, mais c'est tres simple
   ```

6. Définir une fonction d'en-tête `void auteurs_du_livre(char *titre)` qui a pour effet d'afficher, à raison de un par ligne, les noms des auteurs du livre dont le titre est titre. Cette fonction fera appel aux fonctions `lire_livre` et `lire_auteur`. Par exemple, l'appel `auteurs_du_livre("Le langage C, mais c'est très simple")` aura pour effet d'afficher :

   ```
   Claire Dupont
   Pierre Legrand
   ```

7. Définir une fonction `main` qui crée une base de données en utilisant les fonctions `ajouter_livre` et `ajouter_auteur` puis l'interroge en utilisant les fonctions `livres_publies_apres` et `auteur_du_livre`.
8. Tester le programme.

   On prendra soin de traiter les erreurs éventuelles de lecture ou d'écriture dans les fichiers.

Annexe

Fonction `lire_ligne` qui améliore la fonction `fgets` en prenant en compte les deux types de marque de fin de fichier, en tronquant les lignes qui déborderaient du tableau dans lequel elles doivent être stockées et en ne rajoutant pas les marques de fin de ligne à la chaîne de caractères lue. La définition de cette fonction est la suivante :

```
1  char *lire_ligne(char *ligne, int tligne, FILE *f)
2      {
3       char c;
4       int i = 0;
5       while ((c = fgetc(f)) != EOF && c != '\n' && c != '\r')
6         if (i < tligne - 1)
7            ligne[i++] = c;
8       ligne[i] = '\0';
9       if (c == '\r')
10         if ((c = fgetc(f)) == EOF || c != '\n')
11            return NULL;
12      if (c == EOF && i == 0)
13          return NULL;
14      return ligne;
15     }
```

Cette fonction lit la ligne suivante du fichier `f` caractère par caractère. Tant que le caractère lu n'est ni la fin de fichier, ni le caractère `LF` (Line Feed, saut de ligne), ni le caractère `CR` (carriage return, CR, retour à la ligne), il est ajouté à la fin de la chaîne `ligne` si le nombre de caractères lus est inférieur à `tligne - 1` (pas de débordement). A la sortie de cette itération, la fin de la ligne a été atteinte, le caractère nul est ajouté à la chaîne ligne. Si le dernier caractère lu est `CR`, le caractère suivant doit être `LF`, sinon `CR`NULL| est retournée (erreur). Si `EOF` a été lu et qu'aucun caractère n'a été ajouté à la chaîne `s`, `NULL` est retournée (fin de fichier). Dans les autres cas (ligne terminée par `LF` ou `CRLF` ou bien dernière ligne non vide terminée par `EOF`), `ligne` est retournée.