# I2010 : langage C (TP12)

## Les fichiers (binaires)

Reprenez le programme `biblio` écrit lors du TP 9 (solution fournie sur Moodle en semaine 10) et modifiez-le afin de sauvegarder les enregistrements dans un fichier et récupérer un fichier précédent pour ajouter de nouvelles entrées avant de le sauvegarder à nouveau.

L'usage du programme sera :

```
./biblioMaj [fichierIn] fichierOut
```

Si un seul nom de fichier est passé en argument, c'est le nom du fichier `fichierOut` où les enregistrements lus au clavier seront sauvegardés :

```
./biblioMaj fichierOut
```

Si deux noms sont présents, le premier est celui du fichier `fichierIn` qui contient déjà des enregistrements qu'il faudra aller lire et le second celui du fichier `fichierOut` où le programme sauvegardera l'ensemble des enregistrements (i.e. ceux de `fichierIn` et des nouveaux introduits au clavier) :

```
./biblioMaj fichierIn fichierOut
```

Vous pouvez donc tester votre programme à l'aide de la séquence suivante [1] :

`$ ./biblioMaj testIN < bib.txt` (le fichier `testIN` est écrit avec la fonction « ecrireFichier »).

`$ ./biblioMaj testIN testOut < tintin.txt` (le fichier `testIN` est lu avec la fonction « lireFichier » et le fichier `testOUT` est écrit avec « ecrireFichier »).

La structure des enregistrements est la même que celle qui a été définie pour le programme `biblio`, à savoir : le titre, le(s) auteur(s), l'ISBN, l'éditeur, l'année d'édition, le genre (sachant que les genres à prendre en considération pour cette application sont : « BandeDessinée », « Poésie », « Théâtre », « Roman », « RomanHistorique », « LittératureFrançaise », « LittératureEtrangère », « Sciences », « Informatique », « Science-fiction », « Santé », « Histoire »).

Le programme doit :

1. s'il a reçu deux arguments sur la ligne de commande : lire tous les enregistrements de `fichierIn` et les mettre en table ; pour ce faire, ajouter une fonction `lireFichier()` au module *biblio*. Pour faciliter son implémentation, `lireFichier` fait appel à la fonction `ajouterLivre`.

[1] Vous pouvez voir le contenu d'un fichier (binaire) à l'aide de la commande *xxd*, convertisseur hexadécimal de Linux : `xxd testIN`

2. comme le faisait le programme *testBiblio*, tant que la fin de fichier n'est pas atteinte sur l'entrée standard :
    a. lire les informations d'un livre, à raison d'une information par ligne; une ligne vide sépare deux livres
    b. copier les informations dans une structure `Livre`
    c. placer le livre dans la table grâce à la fonction `ajouterLivre` (la taille de la table doit être gérée dynamiquement en fonction du nombre de lignes lues)
3. afficher la table
4. trier la table grâce à la fonction `qsort`, la clé de tri primaire sera l'année d'édition, les clés secondaires seront l'éditeur puis l'auteur
5. afficher la table triée
6. écrire tous les éléments de la table triée dans le fichier dont le nom (`fichierOut`) est passé en argument sur la ligne de commande ; si `fichierOut` n'existe pas, il est créé, sinon il est ouvert en écriture et écrasé ; pour réaliser ce traitement, ajouter une fonction `ecrireFichier()` au module *biblio*.

On suppose que les données lues à l'entrée standard sont conformes au format attendu.

Codes d'erreurs dans le programme principal :

- Si le programme est appelé avec moins d'un argument ou plus de deux, il affiche l'usage sur la sortie d'erreur et s'arrête avec un code d'erreur égal à 1.
- Si le `fichierIn` n'existe pas ou n'est pas ouvrable en lecture, un message d'erreur explicite est affiché à la sortie d'erreur et le programme se termine avec un code d'erreur égal à 10.
- Si le `fichierOut` ne peut être ouvert en écriture, un message d'erreur spécifique est affiché à la sortie d'erreur et le programme se termine avec un code d'erreur égal à 11.
- Si le programme rencontre des problèmes pour lire ou pour écrire dans un fichier, l'utilisateur doit être averti par un message qui précise le type d'erreur avant que le programme ne se termine prématurément avec un code d'erreur à 12 (pour un problème en lecture) ou 13 (pour un problème en écriture).
- Les éventuels problèmes d'allocation de la mémoire doivent aussi être signalés et provoquer l'arrêt du programme avec un code d'erreur à 20.

Pour rappel, la commande Linux permettant d'afficher le code d'erreur renvoyé par le dernier programme exécuté dans le terminal est : `echo $?`