Mini-Projet: Gestion d'une bibliothèque

Ce mini-projet s'étale sur deux séances de TME (séances 3 et 4). Il est noté et doit être rendu sur moodle avant la séance 5 de TME. Les instructions sur la soumission de votre code et de votre rapport sont sur moodle. Dans ce fichier, vous trouverez la partie 1 de ce mini-projet. La partie 2 sera mise sur moodle prochainement.

Dans ce projet, nous nous intéressons à la gestion d'une bibliothèque où une bibliothèque est un ensemble de livres. Un livre est repéré par son titre, le nom de son auteur et un numéro d'enregistrement. Plus précisément, un livre est représenté par les données suivantes :

```
1   int num;
2   char *titre;
3   char *auteur;
```

L'objectif de ce mini-projet est d'apprendre à comparer des structures de données. Plus précisément, nous utiliserons pour implémenter une bibliothèque :

- Une liste simplement chaînée de `struct` (Partie 1).
- Une table de hachage de `struct` (Partie 2).

Vous travaillerez avec la bibliothèque stockée dans le fichier nommé `GdeBiblio.txt` (à télécharger depuis moodle), dont le format est très simple :

- Chaque livre (une entrée) correspond à une ligne.
- Chaque ligne comprend le numéro du livre, suivi de son titre et de son auteur, séparés par un espace.

Pour simplifier la lecture et l'écriture de fichiers, nous supposons dans ce projet que les titres et les auteurs de livres ne contiennent pas d'espaces.

Exercice 1 – Gestion d'une bibliothèque avec une liste chaînée de struct

Dans ce premier exercice, nous allons coder une bibliothèque comme une liste chaînée de `struct` de type livre. On utilisera alors la structure suivante :

```
1  typedef struct livre{
2     int num;
3     char *titre;
4     char *auteur;
5     struct livre * suiv;
6  } Livre;
7
8  typedef struct{ /* Tete fictive */
9     Livre* L;  /* Premier element */
10 } Biblio;
```

Q 1.1 Créez un fichier "biblioLC.h" dans lequel vous ajouterez la définition de cette structure. Dans ce fichier, vous ajouterez au fur et à mesure les signatures des fonctions (opérations) que vous coderez plus tard pour cette structure.

Q 1.2 Créez un fichier "biblioLC.c" dans lequel vous ajouterez progressivement le code des fonctions manipulant cette structure. Commencer par y ajouter les fonctions suivantes :

- `Livre* creer_livre(int num,char* titre,char* auteur)` qui crée un livre.
- `void liberer_livre(Livre* l)` qui réalise une désallocation.
- `Biblio* creer_biblio()` qui crée une bibliothèque vide.
- `void liberer_biblio(Biblio* b)` qui libère la mémoire occupée par une bibliothèque.
- `void inserer_en_tete(Biblio* b,int num,char* titre,char* auteur)` qui ajoute un nouveau livre à la bibliothèque.

Q 1.3 Créez un fichier "entreeSortieLC.h" qui contiendra les signatures des fonctions permettant de manipuler des fichiers. Dans le fichier "entreeSortieLC.c", écrivez les fonctions suivantes :

- `Biblio* charger_n_entrees(char* nomfic, int n, Biblio *b);` permettant de lire n lignes du fichier et de les stocker dans une bibliothèque.
- `void enregistrer_biblio(Biblio *b, char* nomfic);` qui permet de stocker une bibliothèque dans un fichier au bon format : numéro titre auteur.

Q 1.4 Créez un fichier "main.c" contenant une fonction `int main(int argc, char** argv)` permettant à l'utilisateur de donner directement en ligne de commande le nom du fichier contenant la bibliothèque à lire, ainsi que le nombre de lignes à lire dans le fichier. Autrement dit, avec la commande `./main GdeBiblio.txt 100`, le programme doit lire 100 lignes du fichier appelé "GdeBiblio.txt".

Rappel :

- `argv` est un tableau de chaînes de caractères qui est composé d'une chaîne de caractères par mot de la ligne de commande. Dans l'exemple, "./main" compte comme un mot, "GdeBiblio.txt" est un deuxième mot, et "100" est un troisième mot.
- `argc` est le nombre de mots de la ligne de commande. Dans l'exemple, on a trois mots.

`argc` est le nombre de mot de la ligne de commande ("./main" compte comme un mot, "GdeBiblio.txt" est un deuxième mot, et "100" est un troisième mot).

Q 1.5 Créez un fichier `Makefile` et expliquez vos choix dans votre rapport.

Q 1.6 Créez les fonctions suivantes permettant de réaliser :

- l'affichage d'un livre.
- l'affichage d'une bibliothèque.
- la recherche d'un ouvrage par son numéro.
- la recherche d'un ouvrage par son titre.
- la recherche de tous les livres d'un même auteur (retourne une bibliothèque).
- la suppression d'un ouvrage (à partir de son numéro, son auteur et son titre).
- la fusion de deux bibliothèques en ajoutant la deuxième bibliothèque à la première, et en supprimant la deuxième.
- la recherche de tous les ouvrages avec plusieurs exemplaires. Deux ouvrages sont identiques s'ils ont le même auteur et le même titre (seul le numéro change). Cette fonction devra renvoyer une liste comprenant tous les exemplaires de ces ouvrages, avec une complexité-temps pire cas en $O(n^2)$ où n est la taille de la bibliothèque.

Q 1.7 Dans le fichier "main.c", écrire une fonction `void menu()` permettant d'afficher à l'utilisateur toutes les actions possibles sur la bibliothèque. Par exemple : 0-sortie du programme, 1-Affichage, 2-Inserer ouvrage, etc.

Q 1.8 Dans la fonction `main`, faire une boucle qui :

— affiche le `menu` à l'utilisateur,
— récupère l'action à réaliser en lisant sa réponse au clavier,
— puis réalise l'action souhaitée par l'utilisateur.

Cette boucle s'arrête quand l'utilisateur tape '0', indiquant la sortie du programme. On peut par exemple utiliser un `switch case` comme ceci :

```
...
/* Lecture du fichier contenant la bibliotheque (voir question 1.4) */
...
int rep;
do{
    menu();
    scanf("%d",&rep)
    switch (rep){
    case 1:
        printf("Affichage :\n");
        afficher_biblio(B);
        break;
    case 2:
        int num;
        char titre[256];
        char auteur[256];
        printf("Veuillez ecrire le numero, le titre et l'auteur de l'ouvrage.\n")
        /* On suppose que le titre et l'auteur ne contiennent pas d'espace*/
        if (scanf("%d%s%s",&num,titre,auteur)==3){
            inserer_en_tete(B,num,titre,auteur);
            printf("Ajout fait.\n")
        }else{
            printf("Erreur format\n");
        }
        break;

        etc.
    }
}while(rep!=0);
printf("Merci, et au revoir.\n")
return 0;
```

Attention : Dans le cours, nous avons vu qu'il était préférable d'utiliser la fonction `fgets` plutôt que `scanf`. Nous vous demandons donc d'utiliser la première fonction plutôt que la deuxième.