LU2IN006 – Structure de Données

**RAPPORT : MINI-PROJET 2**

BOUAMEUR Lina

CHARPENTIER Tristan

Groupe 2

**Exercice 1 et 2 : Les fonctions communes :**

1. **FONCTIONS MANIPULANT LES FICHIERS.TXT**

Toutes les définitions de ces fonctions peuvent être retrouvées dans entreeSortieLC.h et entreeSortieH.h !

Les deux exercices traitent un fichier.txt, du coup nous avons créer deux fonctions :

1. **charger\_n\_entrees** : Permettant de créer une bibliothèque (aussi bien pour les listes chaînées BiblioLC\* que pour la table de hachage BiblioH\*) qui regroupera les entrées du fichiers .txt. Pour cela, nous devrons manipuler le fichier grâce aux fonctions ***fopen, fgets,sscanf*** et ***fclose****.*
2. **Enregistrer\_biblio :** Permettant d’enregistrer des livres dans une nouvelle bibliothèque (aussi bien pour les listes chaînées BiblioLC\* que pour la table de hachage BiblioH\*) grâce aux fonctions ***fopen,fprintf*** et ***fclose***

*A quoi servent ces fonctions ?*

fopen : permet d’ouvrir un fichier.

fclose : permet de fermer un fichier.

fgets : permet la lecture du fichier (ici on fait ligne par ligne)

fprintf : permet d’écrire une chaîne dans un fichier.

1. **FONCTIONS MANIPULANT LA MEMOIRE (MALLOC ET FREE).**

Toutes les définitions de ces fonctions peuvent être retrouvées dans biblioLC.h et biblioH.h !!

1. **Livre\*/LivreH\* créer\_livre (int num, char \*titre, char \*auteur)** : Pour créer un livre nous devons d’abord réserver un espace pour le nouveau livre grâce à un malloc pour les structures Livre\* ou LivreH\*. Nous allons ensuite affecter à ce nouveau livre les champs num, titre et auteur. Nous affectons NULL à l->suiv puisqu’on ne déclare qu’un seul livre, il n’y a donc aucun livre qui le suit. Nous retournons à la fin le livre.
2. **Void libérer\_livre (Livre\*/LivreH\* l)** : Après l’utilisation d’un malloc il est préférable de libérer l’espace utilisée, c’est ce que nous faisons ici grâce à la fonction free.
3. **Biblio\*/BiblioH créer\_biblio (void/int m)**: Il s’agit exactement du même cas que créer\_livre sauf que nous créons une bibliothèque de structures. Il y a quelques différences entre les listes chaînées et les tables de hachage que nous évoquerons dans le résumé des exercices.
4. **Void liberer\_biblio (Biblio\*/BiblioH\* b)**: Comme pour livre nous allons libérer la bibliothèque grâce à un free. Cependant la complexité réside dans le fait que l’on doit dabord free chaque livre inclut dans la bibliothèque avant de free la bibliothèque en entier d’où l’intérêt d’utiliser une boucle.
5. **FONCTIONS D’INSERTION ET DE FUSION :**

Toutes les définitions de ces fonctions peuvent être retrouvées dans biblioLC.h et biblioH.h !!

1. **void inserer\_en\_tete(Biblio\* B, int num, char\* titre, char\* auteur) :** On crée tout d’abord un nouveau livre grâce à notre fonction créer\_livre (pas besoin d’utiliser un malloc puisqu’il est déjà inclut dans créer\_livre),le livre suivant notre nouveau livre sera donc le premier ancien livre en tête, on lui donne alors la position de nouveau livre suivant, on remplace ensuite l’entête par le nouveau livre. \*Attention le même raisonnement est utilisé pour hachage à différence d’utilisation d’une clé et d’une fonction d’hachage. \*
2. **void fusion(Biblio\*/BiblioH \* b1, Biblio\*/BiblioH\* b2**) : Des commentaires ont été utilisés dans le programme pour expliquer chaques étapes, en général on parcourt la bibliothèque b1 en entier, puis une fois qu’on atteint le dernier livre, on introduit notre deuxième bibliothèque, que l’on supprimera à la toute fin grâce à un free().
3. **FONCTIONS DE RECHERCHE :**

Toutes les définitions de ces fonctions peuvent être retrouvées dans biblioLC.h et biblioH.h !!

La majorité de ces fonctions consiste à parcourir la bibliothèque jusqu’à ce qu’une condition est réalisé (if), puis il suffit d’afficher le résultat.

**Exercice 1 : Gestion d’une bibliothèque avec liste chaînée.**

Cette partie contient majoritairement du code.

Nous allons créer deux bibliothèques (biblioLC.h et entreeSortieLC.h) qui comprennent l’ensemble des fonctions de biblioLC.c et entreeSortieLC.c. Nous allons ensuite tester ces deux fonctions dans un main en proposant un choix à l’utilisateur grâce à une fonction menu ().

Pour pouvoir compiler toutes ces fonctions de manière facile et rapide, nous avons aussi créer un Makefile.

**Structures utilisées :**

typedef struct livre {

    int num;

    char \*titre;

    char \*auteur;

    struct livre \* suiv;

} Livre;

typedef struct { /\* Tete fictive \*/

    Livre\* L; /\* Premier element \*/

} Biblio;

**Exercice 2 : Gestion d’une bibliothèque avec une table de hachage.**

Cette partie contient majoritairement du code.

Nous allons créer deux bibliothèques (biblioH.h et entreeSortieH.h) qui comprennent l’ensemble des fonctions de biblioH.c et entreeSortieH.c. Nous allons ensuite tester ces deux fonctions dans un main en proposant un choix à l’utilisateur grâce à une fonction menu ().

Pour pouvoir compiler toutes ces fonctions de manière facile et rapide, nous avons aussi créer un MakefileH.

**Structures utilisées :**

typedef struct livreh {

    int clef;

    int num;

    char\* titre;

    char\* auteur;

    struct livreh \*suiv;

} LivreH;

typedef struct table {

    int nE; /\*nombre d'éléments contenus dans la table de hachage \*/

    int m; /\*taille de la table de hachage \*/

    LivreH\*\* T; /\*table de hachage avec résolution des collisions par chainage \*/

} BiblioH;

**Les différences :**

Deux nouvelles fonctions sont introduites ici : fonctionClef et fonctionHachage.

1. **Int fonctionClef (char\* auteur) :** Cette fonction permet d’attribuer une clé à un livre. Pour réaliser cela, nous effectuerons une boucle qui parcourra l’ensemble de la chaîne auteur et qui renverra la somme de chaque caractère. Nous avons alors une clé.
2. **Int fonctionHachage (int cle, int m) :** Cette fonction permet de stocker les livres dans notre table grâce à leur clé. Elle est très pratique puisqu’elle permet d’éviter au maximum les collisions. Ici notre équation est la suivante : h(k)= m\*(clé\*A – floor(cle\*A)) \*On utilise floor ici dans le but d’éviter une fonction retournant un 0, puisque clé\*a-clé\*A donnerai 0\*. Ici A représente le chiffre d’or.

**\*Puisqu’on a besoin d’une fonction de hachage pour stocker un livre, alors l’insertion du livre dans une bibliothèque se fera forcément grâce à notre fonction hachage, d’où la différence mentionnée précédemment dans *exercice 1 et 2 : fonctions communes (III)*.**

**Exercice 3 : Comparaison de deux structures.**