R. Malgouyres, R. Zrour et F. Feschet
Initiation à l'algorithmique et à la programmation en C,
Cours avec 129 exercices corrigés,
DUNOD, Collection Sciences Sup, 2011, 2^e édition

Algorithmique et programmation en C

TP n° 7
Fichiers binaires

durée : 2 semaines

Objectifs :
Le but du TP est de se familiariser avec fichiers binaires. On implémentera une base de données en la chargeant en mémoire. On implémentera le `main` dans un fichier différent des autres fonctions.

1 La base de données d'une librarie

Dans la base de données d'une librairie, chaque livre est représenté par une structure :

```
typedef struct{
              int numero;          /* référence article */
              char auteur[30];  /* nom de l'auteur */
              char editeur[30]  /* nom de l'éditeur */
              char titre[100];  /* titre */
              int stock;    /* quantité disponible en stock */
             }TypeLivre;
```

La base de donnée est stockée dans un fichier binaire au format suivant :

1. La première donnée du fichier est un `int` contenant le nombre total de livres de la base de données ;
2. Suivent toutes les structures représentant tous les livres de la base. **On suppose que les numéros des livres se suivent et correspondent à leur position dans le fichier**.

2 Travail à effectuer

Pour chaque exercice, on ajoutera une option à un menu dans le programme principal qui fera appel à la fonction crées pour l'exercice. Le programme principal sera implémenté dans un fichier `main.c` différent du fichier source contenant les autres fonctions et on fera un `makefile`.

Exercice 1 Écrire une fonction qui crée en mémoire une base de donnée avec 0 livres dans un fichier dont le nom est passé en paramètre. On prévoiera un espace mémoire initial pour 100 livres. Une variable `nbmax` du `main` contiendra la valeur 100 et pourra être modifiée par la suite par une autre fonction.

Exercice 2 Écrire une fonction de prototype

```
TypeLivre* CreeLivre(TypeLivre* ancientab, int *pnb, int *pnbmax)
```

qui permet à un utilisateur de rajouter un livre dans la base de données. Le fonction incrémentera le nombre de livre. Si ce nombre devient trop grand, la fonction réalloue le tableau (alloue un tableau plus grand et recopie les données existantes, par exemple en ajoutant 100 places en mémoire), augmente la variable `nbmax` du `main`, et retourne le nouveau tableau.

Exercice 3 Écrire une fonction de sauvegarde dans un fichier binaire et une fonction de chargement de la base de données en mémoire.

Exercice 4 Écrire une fonction de prototype

```
void AccesBdd(TypeLivre *tab, int nb, int numLivre, TypeLivre *plivre);
```

qui renvoie les données d'un livre (par passage par adresse) de la base de données à partir de son numéro.

Exercice 5 Écrire une fonction d'affichage d'un livre. En l'utilisant, testez la fonction `AccesBdd`.

Exercice 6 Écrire une fonction de prototype

```
void ModifieLivre(TypeLivre *tab, int nb, int numLivre);
```

qui permet à un utilisateur de modifier les données d'un livre à partir de son numéro.

Exercice 7 Écrire fonction de prototype

```
char CommandeLivre(TypeLivre *tab, int nb, int numLivre, int quantité)
```

qui met à jour les stocks lors de la commande d'une certaine quantité d'un livre par un client. La fonction doit renvoyer 1 en cas d'erreur et 0 sinon.

Exercice 8 Écrire une fonction de prototype

```
int RechercheTitre(FILE *fp, char* titre);
```

qui renvoie le numéro d'un livre à partir de son titre en recherchant dans la base.

Exercice 9 En utilisant la fonction de l'exercice 8 et la fonction de l'exercice 4, Écrire une fonction de prototype

```
int AfficheLivre(FILE *fp, char* titre);
```

qui permet à un utilisateur d'afficher les données d'un livre à partir de son titre.

Exercice 10 Écrire une fonction qui affiche tous les livres d'un auteur donné.

Exercice 11 Écrire une fonction qui affiche tous les livres dont le titre contient un mot passé en paramètre. On pourra utiliser la fonction suivante de la bibliothèque `string.h` :

```
char* strstr(char* chaine, char* mot);
```

qui recherche une sous-chaîne `mot` dans une chaîne `chaine` et renvoie `NULL` si la chaîne `mot` n'apparaît pas dans la chaîne `chaine`.