TP n° 2 : Gestion des Fichiers

2015-2016

Objectif *:*

- Manipulation des **fichiers** côté utilisateur
- Opérations de base sur les fichiers **textes** et **binaires**

1. Fichier texte

On désire maintenir un fichier de clients dans une agence bancaire. Un client est référencé par :

- Son code unique : `int code`
- Son nom : `char nom[80]`
- Son solde : `double solde`

a. Créer par programme un fichier **texte** «`clients.txt`» pour sauvegarder 2 clients dont les données seront lues au clavier. Utiliser la fonction d'E/S : **`fprintf()`**

b. Editer le fichier à l'aide d'un **éditeur de texte**.

c. Afficher par programme les noms des 2 clients. Utiliser la fonction d'E/S : **`fscanf()`**

2. Fichier binaire

Dans le cadre de la réalisation d'un jeu on désire sauvegarder la partie d'un joueur sur fichier **binaire**. On suppose que le joueur a un **nom**, se trouve dans un **lieu** qui a un numéro (entre 0 et 50) et possède une **liste d'objets** (10 objets au total). Par exemple :

```
num_lieu=39 ;      //le joueur se trouve dans le lieu n° 39
liste_objet[3]=1;  //le joueur possède l'objet 3
liste_objet[7]=1;  //le joueur possède l'objet 7  … etc
```

a. Compléter la fonction **`sauvegarder_jeu()`** qui permet de créer, par programme, un fichier **binaire** pour y sauvegarder les informations du joueur. Utiliser la primitive **`fwrite()`** pour l'écriture dans le fichier.

```
void sauvegarder_jeu(char *nom_partie){
  FILE      *flot;
  //OUVERTURE FICHIER (VERIFICATION D'ERREUR)
  flot =fopen(nom_partie, "wb");
    /* ECRITURE NUMERO DU LIEU */
  fwrite (&num_lieu, sizeof(num_lieu),  1, flot)
    /* ECRITURE LISTE DES OBJETS */
  fwrite (liste_objet, sizeof(liste_objet),1, flot);
    /* FERMETURE FICHIER */
  fclose (flot);
}
```

b. Tester cette fonction en renseignant d'avance les données du joueur (**`num_lieu`** et **`liste_objet`**).

c. Editer le fichier à l'aide d'un éditeur de texte. Que remarquez-vous ?

d. Récupérer par programme les informations concernant le joueur à partir du disque (et les lister). Ecrire pour cela une fonction `void` **`charger_jeu`**`(char *nom_partie)` et utiliser la primitive **`fread()`** pour la lecture des données.

3. Sauvegarde des adhérents sur disque (.\adher.dta)

Nous voulons maintenir un fichier des adhérents d'une association scientifique (telle que vue en TP2). Les adhérents sont répartis sur plusieurs villes du royaume.
Les opérations **d'insertion** des adhérents, **d'affichage** ou de **suppression** seront réalisées sur une structure de donnée au niveau de la mémoire centrale (tableau de listes) et non directement sur disque.

- A l'ouverture de la session (lancement du programme) on doit **`charger()`** automatiquement les adhérents en **mémoire**.
- A la fin de la session (fin du programme) on doit **`sauvegarder()`** automatiquement les adhérents sur **disque**.

Le fichier «**`adher.dta`**» est **binaire** et est structuré de la façon suivante :

$$n\ adh_1\ adh_2\ \dots\ adh_n$$

On y trouve d'abord le nombre total des adhérents (n) ensuite la liste de tous les adhérents.

- Écrire cette application en C en proposant un **`menu()`** à l'utilisateur

Annexe

Programmation modulaire :

- Séparer, dans le même projet, un fichier d'entête qui ***déclare*** les types de données et fonctions sans les définir : **declar.h**
- ***Définir*** ces fonctions dans un autre fichier : **fonctions.c**
- Le programme de ***test***, main(), doit être dans un fichier : **test.c**

Fonctions d'E/S :

```
int fread(void * destination, int tailleElement, int nombreElts, FILE *flot);

int fwrite( const void *source, int tailleElement, int nombre, FILE *flot);
```