SÉANCE 11

Objectif

Le but de cette séance est de manipuler des tableaux de structures, d'effectuer des entrées-sorties dans des fichiers binaires et d'utiliser un pointeur de fonction lors de l'appel de la fonction générique `qsort`.

Outils

Voici une fonction de la bibliothèque standard de manipulation des chaînes de caractères qui vous sera utile lors de cette séance. Elle nécessite l'inclusion de `string.h`.

```
int strcmp(const char *Ch1, const char *Ch2)
```

La fonction `strcmp` retourne :

- 0 si les deux chaînes sont identiques ;
- un entier positif si la chaîne représentée par `Ch1` est supérieure à la chaîne représentée par `Ch2` selon l'ordre lexicographique ;
- un entier négatif si la chaîne représentée par `Ch1` est inférieure à la chaîne représentée par `Ch2` selon l'ordre lexicographique.

Exercice

Exercice (Gestion d'un répertoire de contacts : version statique)

Récupérez le fichier `tp11ex.c` afin de compléter son contenu. Dans ce fichier :

- un contact est représenté par une structure `sContact` contenant quatre champs correspondant au nom, au prénom, à l'adresse mail et à la date de naissance ;
- une date est représentée par une structure contenant trois champs correspondant au jour, au mois et à l'année ;
- un répertoire de contacts est représenté par un tableau statique de contacts.

On suppose que :

- un répertoire contient au plus 64 contacts ;
- un nom est une chaîne d'au plus 30 caractères ;
- un prénom est une chaîne d'au plus 20 caractères ;
- une adresse mail est une chaîne d'au plus 254 caractères.

Suivez les indications placées en commentaires dans le fichier source `tp11ex.c` et écrivez le corps des fonctions suivantes (pour les tester au fur et à mesure, une fonction `main` a été écrite) :

1. `void AfficherContact2(const struct sContact *pContact)`
   qui affiche à l'écran les valeurs des champs de la variable structurée pointée par `pContact` (le qualifieur `const` permet de préciser aux utilisateurs de la fonction `AfficherContact2` que cette dernière ne modifiera pas la variable structurée pointée). Inspirez-vous de la fonction `AfficherContact1` dont le corps est présent dans le fichier et qui prend en entrée la variable structurée, pas son adresse.
2. `void AfficherRepertoire(struct sContact Repertoire[], int NbContacts)`
   qui affiche à l'écran le contenu du répertoire `Repertoire` contenant `NbContacts` personnes. Les informations de chaque personne seront affichées en faisant appel à la fonction `AfficherContact2` de la question 1.

3. `void AjouterContact(struct sContact *pNouveau, struct sContact Repertoire[], int *pNbContacts)`
   qui ajoute le contact pointé par `pNouveau` à la fin du répertoire `Repertoire` et met à jour le nombre de contacts pointé par `pNbContacts`.
4. `int Rechercher(char NomRecherche[], struct sContact Repertoire[], int NbContacts)`
   qui recherche dans le répertoire `Repertoire` contenant `NbContacts` la première personne dont le nom est identique à celui contenu dans la chaîne de caractères `NomRecherche`. Cette fonction doit retourner :
   - l'indice de l'élément trouvé dans le tableau `Repertoire` si le nom existe dans le répertoire ;
   - `-1` sinon.

   Cette fonction utilisera la fonction `strcmp` de la bibliothèque standard.
5. `void EcrireFichier(struct sContact Repertoire[], int NbContacts, char NomFichier[])`
   qui écrit au format binaire les `NbContacts` contacts du répertoire `Repertoire` dans le fichier de nom `NomFichier`. Les contacts étant stockés dans un tableau, cette écriture peut se faire avec un seul appel de la fonction `fwrite`.
6. `int LireFichier1(struct sContact Repertoire[], char NomFichier[])`
   qui lit dans le fichier binaire de nom `NomFichier` un répertoire, le stocke dans le tableau `Repertoire` et retourne le nombre de contacts lus. La lecture se fera contact par contact, c'est-à-dire par une répétition d'appels à la fonction `fread` jusqu'à que tous les contacts aient été lus.
7. `int LireFichier2(struct sContact Repertoire[], char NomFichier[])`
   qui fait la même chose que la fonction précédente mais, après avoir calculé le nombre de contacts présents dans le fichier, effectue la lecture par un seul appel de la fonction `fread`.
8. `int CompareContacts(const void *pc1, const void *pc2)`
   qui compare les deux noms des deux contacts pointés par `pc1` et `pc2`, et retourne un entier supérieur, égal ou inférieur à 0, selon que le nom du premier contact est supérieur, égal ou inférieur au nom du second contact, dans l'ordre lexicographique. Cette fonction utilisera la fonction `strcmp` de la bibliothèque standard. Dans la fonction `main`, écrivez l'appel à la fonction `qsort` permettant de trier les contacts d'un répertoire en utilisant la fonction `CompareContacts`.