SÉANCE 12

Objectif

Le but de cette séance est de manipuler une structure de données dynamique : la liste simplement chaînée.

Exercice

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

Cet exercice consiste à modifier le corps de la plupart des fonctions que vous avez écrites lors de la précédente séance afin de prendre en compte une autre structure de données. Récupérez le fichier `tp12ex.c` afin de compléter son contenu. Dans ce fichier :

- un contact est représenté par une structure `sContact` contenant cinq champs correspondant au nom, au prénom, à l'adresse mail, à la date de naissance et à l'adresse du contact suivant ;
- 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 une liste simplement chaînée de contacts à laquelle on accède par une structure contenant l'adresse du premier contact et l'adresse du dernier contact.

On suppose que :

- 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.

En n'oubliant pas de représenter schématiquement la mémoire, suivez les indications placées en commentaires dans le fichier source `tp12ex.c` et écrivez le corps des fonctions suivantes (pour les tester au fur et à mesure, une fonction `main` a été écrite) :

1. `tRepertoire CreerRepertoire(void)`
   qui crée et retourne un répertoire vide.
2. `void AfficherRepertoire(tRepertoire Repertoire)`
   qui affiche à l'écran le contenu du répertoire `Repertoire`.
3. `void AjouterContact(struct sContact *pNouveau, tRepertoire Repertoire)`
   qui ajoute le contact pointé par `pNouveau` à la fin du répertoire `Repertoire`.
4. `struct sContact *Rechercher(char NomRecherche[], tRepertoire Repertoire)`
   qui recherche dans le répertoire `Repertoire` la première personne dont le nom est identique à celui contenu dans la chaîne de caractères `NomRecherche`. Cette fonction doit retourner :
   - l'adresse du contact trouvé si le nom existe dans le répertoire ;
   - `NULL` sinon.
5. `void EcrireFichier(tRepertoire Repertoire, char NomFichier[])`
   qui écrit au format binaire les contacts du répertoire `Repertoire` dans le fichier de nom `NomFichier`.
6. `tRepertoire LireFichier(char NomFichier[])`
   qui lit dans le fichier binaire de nom `NomFichier` un répertoire et retourne l'accès au répertoire créé ou `NULL` en cas de problème.
7. `void LibererRepertoire(tRepertoire *pRepertoire)`
   qui libère tout l'espace mémoire occupé par un répertoire et affecte `NULL` à l'accès au répertoire pointé par `pRepertoire`.