Travaux Pratiques d'algorithmique n°6

Cours d'Informatique de Deuxième Année

—DEUG Sciences M3-MI—

Listes chaînées

Le but de ce TP est de gérer un petit répertoire d'étudiants grâce aux listes chaînées. Les principales opérations à réaliser seront les suivantes :

- la saisie des coordonnées d'un étudiant ;
- l'ajout d'un étudiant dans le répertoire ;
- la suppression d'un étudiant du répertoire ;
- la visualisation et le tri du répertoire.

▸ **Exercice 1. (saisie des coordonnées d'un étudiant)** *Un étudiant est caractérisé par les informations suivantes :*

- *son nom ;*
- *son prénom ;*
- *son numéro de TP ;*
- *son numéro de TD.*

On définit le type `Fiche` *approprié :*

```
struct etudiant
       {
         char * nom;
         char * prenom;
         unsigned int tpno;
         unsigned int tdno;
       };

typedef struct etudiant Fiche;
```

Ecrire les fonctions `void SaisirFiche(Fiche *f)` *et* `void AfficherFiche(Fiche f)` *pour respectivement, saisir et afficher les coordonnées d'un étudiant.*

▸ **Exercice 2. (Gestion du répertoire)** *On souhaite gérer le répertoire sous forme d'une liste chaînée de "fiches" d'étudiants. Le type* `Repertoire` *est défini par :*

```
struct repert
       {
         Fiche         fiche;
         struct repert *suiv;
       };

typedef struct repert Repertoire;
```

Ecrire les fonctions :

- `Repertoire * AllouerFiche(Repertoire * rep)` *pour allouer une fiche en tête de liste du répertoire, (retour* `NULL` *si l'allocation mémoire est impossible) ;*
- `void LibererFiche(Repertoire *)` *qui libère l'espace mémoire consommé par la fiche située en tête de liste du répertoire.*

▸ **Exercice 3.** *Ecrire les fonctions :*

- `int FichesIdentiques(Fiche f1, Fiche f2)` *qui teste l'égalité des deux fiches* `f1` *et* `f2` *;*
- `int MembreRepert(Repertoire *rep, Fiche f)` *qui indique si la fiche* `f` *est présente dans le repertoire* `rep` *(retourne 1 si vrai, 0 sinon).*

▸ **Exercice 4. (Gestion d'un étudiant dans le répertoire)** *Ecrire les fonctions suivantes :*

- `Repertoire * AjouterFiche(Repertoire *rep, Fiche f)` *;*
- `Repertoire * SupprimerFiche(Repertoire *rep, Fiche f)` *;*
- `void AfficherRepertoire(Repertoire *rep)` *;*
- `Repertoire * DetruireRepertoire(Repertoire *rep)`.

▸ **Exercice 5. (Tri du répertoire)** *En vous servant des algorithmes de tri que vous avez appris, écrivez une fonction* `TrierRepertoire` *de tri performante du répertoire (en fonction du champ* `nom`*).*

▸ **Exercice 6. (Rechercher un étudiant dans le répertoire)** *Ecrire la fonction* `void FiltreRepertoire(Repertoire *rep, Fiche crit)` *qui permet de rechercher un (ou plusieurs) étudiant(s) en fonction de ses caractéristiques (son nom, son prénom, son numéro de TP, son numéro de TD). Une manière de faire est de réutiliser le module de saisie des coordonnées d'un étudiant. Les informations saisies sont alors les critères de recherche. Si un champ de saisie n'est pas un critère de recherche, alors il suffira de taper sur retour chariot (chaîne vide pour le nom ou le prénom) ou bien de rentrer -1 (pour le numéro de TD ou TP). On devra pouvoir* ***visualiser*** *simplement les réponses multiples.*

▸ **Exercice 7. (Gestions des notes)** *On veut maintenant gérer les notes des étudiants. Reprendre l'exercice précédent avec les champs de données suivants :*

- *son nom ;*
- *son prénom ;*
- *son numéro de TP ;*
- *son numéro de TD ;*
- *ses notes de TP ;*
- *ses notes de TD.*

Quelle structure de données proposez-vous pour modéliser les notes ? Pourquoi ? On veut pouvoir saisir la note d'un TP d'un étudiant donné à n'importe quel moment. Ecrire deux fonctions permettant d'ajouter `AjouterNote` et de modifier `ModifierNote` des notes dynamiquement. Faire un menu général permettant de saisir les informations relatives à un étudiant, ses notes obtenues et de calculer la moyenne de toutes les notes disponibles au moment du calcul. Comment faire pour gérer les absences aux TD ou TP et obtenir des moyennes correctes ?