Série N° 6 : Les pointeurs et les listes

EXERCICE 1 :

Soit une matrice (N,M) d'entiers. Ecrire un algorithme qui génère deux listes à partir de cette matrice.

1- La première regroupe les maximums des lignes (FIFO);
2- Et, la deuxième la somme des colonnes (LIFO).

EXERCICE 2 :

Soit une liste d'entiers **L**, écrire les actions paramétrées suivantes permettant :

1- La suppression des doublons (éléments identiques) ;
2- La suppression de la valeur minimale d'une liste;
3- La création de la liste miroir de **L** (avec ensuite sans création d'une nouvelle liste) ;
4- La fusion de deux listes triées d'entiers **L1** et **L2** en une liste triée **L3** (avec ensuite sans création d'une nouvelle liste);

EXERCICE 3 :

Soit **L** une liste d'entiers positifs. Ecrire une procédure qui permet d'éclater la liste **L** en deux listes : **Lp** contenant les entiers positifs et **Ln** contenant les entiers négatifs. (Sans création de nouvelles listes).

EXERCICE 4 :

Soient deux listes **L1** et **L2** :

1- Ecrire une fonction qui vérifie si **L1** et **L2** sont identiques (contiennent les mêmes éléments dans le même ordre),
2- Ecrire une fonction qui vérifie si **L1** est incluse dans **L2** (tous les éléments de **L1** se trouvent dans **L2**, ici l'ordre ne compte pas),
3- Ecrire une fonction qui vérifie si **L1** est une sous-liste de **L2 (L1** est incluse dans **L2** dans le même ordre**).**

EXERCICE 5 :

On veut construire un mot à base des trois lettres A, B et C en respectant les règles suivantes :
Au début (à l'étape 0) le mot est réduit à A. Puis à chaque étape de l'évolution, A est transformé en ABC, B est transformé en C, et C est transformé en A.

Exemple : Etape 0 Etape 1 Etape 2 Etape 3

A → A B C → A B C C A → A B C C A A A B C → …

On veut connaître le mot à l'étape N, N étant donné. En utilisant une liste chaînée, dont chaque cellule contient une lettre (A ou Bou C).

1- Ecrire une procédure permettant d'afficher les éléments d'une liste de caractères.
2- Ecrire un algorithme qui suit l'évolution du mot et affiche le mot construit à chaque étape.

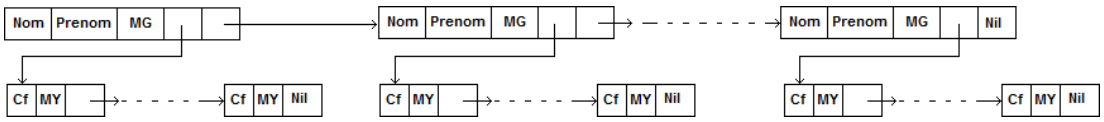
EXERCICE 6 :

Soit **FENT** un fichier d'entiers.

1- Ecrire une action paramétrée **FLISTE** permettant de créer une liste chainée **L** contenant les éléments de **FENT** dans le même ordre (**en FIFO**).
2- Soit **VAL** une valeur entière donnée. Ecrire une action paramétrée permettant de créer une liste **POSVAL** contenant toutes les positions de la valeur **VAL** dans la liste **L** (**en LIFO**).
3- Soit **K** un entier positif. Ecrire une action paramétrée **DELPOS** permettant de supprimer l'élément se trouvant à la **K**eme position de la liste **L**.

EXERCICE 7:

Soit L une liste d'étudiants. Chaque étudiant est défini par son **Nom**, son **Prénom** et sa moyenne générale **MG**. Chaque élément de la liste contient un pointeur vers une liste des notes de l'étudiant. Un élément de la liste des notes contient un coefficient **Cf** et une moyenne **MY.**



1- Donner la déclaration des deux listes (**Letud, Lnote**).
2- Ecrire une AP **Moyenne** permettant de calculer la moyenne d'une liste de notes.
3- Ecrire une AP **Delib** permettant de calculer la moyenne générale de chaque étudiant.
4- Ecrire une AP **Supprime** permettant de supprimer les étudiants ayant une MG<5.
5- Ecrire une AP **Resultat** permettant de créer (**sans allocation**) deux listes **Ladm** et **Lajn** contenant respectivement les étudiants admis et les étudiants ajournés.

EXERCICE 8:

Soit **L** une liste d'entiers. Ecrire une fonction qui renvoie l'élément se trouvant au **milieu** de la liste en utilisant un **seul parcours**.

Série Complémentaire

EXERCICE 9 :

Soit T un tableau de 26 listes de chaînes de caractères. La liste 1 contient des mots commençant par la lettre 'A', la liste 2 contient des mots commençant par la lettre 'B'…etc.
Déclarer T et écrire une fonction qui vérifie l'existence d'un mot M dans la structure.

EXERCICE 10 :

Soient deux listes (L1 et L2) de valeurs entières positives :

1) Donner les déclarations des listes ;
2) Ecrire une action paramétrée permettant de déplacer (sans allocation ni libération) les valeurs paires de L1 vers L2, et, de déplacer les valeurs impaires de L2 vers L1 ;

EXERCICE 11 :

Un pharmacien souhaite traiter les informations concernant son stock de médicaments par ordinateur. On vous propose de représenter ces informations sous forme de liste linéaire chainée où chaque élément contient le libellé d'un médicament, la quantité disponible (nombre de boites) et le prix unitaire.

1. Donner les structures de données nécessaires a la représentation de ce stock (voir schéma).
2. Ecrire la procédure **Vendre** (Med, NbBoites) permettant de retirer, si possible, 'NbBoites' du médicament 'Med' du stock. (Il faut supprimer du stock le médicament dont la quantité atteint 0).
3. Ecrire la procédure **Acheter** (Med, NbBoites, Prix) permettant au pharmacien d'alimenter son stock par 'NbBoites' du médicament 'Med' ayant le prix unitaire 'Prix' DA. On considère qu'un médicament prenne toujours lenouveau prix. Si le médicament n'existe pas, il faut l'insérer.
4. Ecrire la fonction **ValStock** permettant de calculer la valeur des médicaments dans le stock.

EXERCICE 12 :

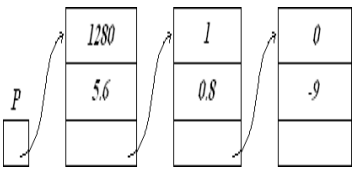
Soit L une liste de caractères constituant des mots (un mot est une suite de caractères ne contenant pas de blanc) séparés par un seul caractère blanc (espace).
Ecrire une procédure qui inverse les mots de la liste L sans création d'une nouvelle liste.

EXERCICE 13 :

Dans cet exercice, on se propose de développer un module permettant de manipuler des polynômes creux. Un polynôme creux est un polynôme contenant très peu de monômes non nuls.
Exemple : $P(x) = 5.6\ x^{1280} + 0.8\ x - 9$ contient 1281 termes dont 3 seulement sont non nuls.
Chaque monôme est décrit par un enregistrement de type *Tmonome* comportant les 3 champs suivants :

- *Deg* : entier représentant le degré du monôme ;
- *Coef* : réel représentant le coefficient du monôme ;
- *Suiv* : pointeur sur le monôme suivant.

La liste représentant le polynôme sera triée par ordre de degré décroissant. Une liste vide (Nil) correspond au polynôme zéro ;



1- Ecrire une procédure **Ajouter** qui ajoute à un polynôme (Pol) la valeur d'un monôme défini par son degré (Deg) et son coefficient (Coef).
2- Ecrire les procédures **Somme** et **Produit** qui réalisent respectivement la somme et le produit de deux polynômes Pol1 et Pol2.
3- Ecrire une fonction **Valeur** qui calcule la valeur du polynôme pour une valeur val de la variable x.
4- Ecrire une procédure **Derive** qui détermine la dérivée DPol d'un polynôme Pol.
5- Ecrire une procédure **Primitive** qui détermine la primitive PPol d'un polynôme Pol, sachant que PPol(0)=1.

EXERCICE 14 :

Soit L une liste d'entiers.

1- Ecrire une procédure **DETACHE** qui renvoie l'adresse du minimum de la liste L et le détache de la liste sans le supprimer.
2- En utilisant la procédure **DETACHE**, écrire une procédure **TRIER** qui trie la liste L dans un ordre décroissant (sans création de nouvelle liste).

EXERCICE 15 :

Soient deux listes **L1** et **L2** :

1- Ecrire une fonction qui vérifie si **L1** et **L2** sont disjointe (**L1** ∩ **L2** = Ø).
2- Ecrire une fonction qui vérifie si **L1** est préfixe de **L2 (L2** commence par **L1).**