– Algorithmique – TD N°5 – Les Listes Linéaires Chaînées (LLC)

Dans tous les exercices on utilise des listes d'entiers.

Rappel de cours

Le type **Liste** est déclaré comme suit :

```
Type   Liste = ^Element
       Element = Enregistrement
           | Val : Entier
           | Suiv : Liste
       Fin
Var L, P : Liste /* Deux variables de type Liste */
```

Opérations sur les pointeurs

- ***Initialisation :*** On écrit P ← Nil. ***Nil :*** est une constante qui représente une adresse particulière qui ne pointe vers aucune donnée.
- ***Allouer(P)*** : permet de créer un nouvel élément (ou cellule) pointé par le pointeur ***P***.
- ***Liberer(P)*** : permet de supprimer l'élément (ou la cellule) pointé par le pointeur ***P.***

Exercice 1 : Construction et exploitation

Q1) Écrire une procédure ***InsererTete(L, N)*** qui permet d'insérer l'entier ***N*** au début de la liste ***L*** (La liste ***L*** peut être vide).

Q2) Utiliser la procédure ***InsererTete*** pour écrire une autre procédure ***CreerListe*** qui permet de créer une liste d'entiers positifs à partir du clavier. Si une valeur négative est saisie, l'insertion à la liste s'arrête.

Q3) Écrire une procédure ***AfficherListe(L)*** qui permet d'afficher les éléments de la liste ***L***. Si ***L*** est vide, la procédure doit afficher le message '*** Liste vide ***'.

Q4) Écrire l'algorithme principal qui permet de tester les procédures écrites dans les questions précédentes.

Q5) Écrire une procédure ***InsererQueue(L, N)*** qui permet d'insérer l'entier ***N*** à la fin de la liste ***L*** (La liste ***L*** peut être vide).

*** Q6)** Modifier la procédure écrite à la question **Q2** pour utiliser la procédure ***InsererQueue***.

Q7) Écrire une fonction ***Longueur(L)*** qui permet de calculer la longueur (nombre d'éléments) de la liste ***L***.

Q8) Écrire une procédure récursive ***AfficherListeREC(L)*** qui permet d'afficher les éléments de la liste ***L***.

Q9) Modifier la procédure ***AfficherListeREC(L)*** pour qu'elle affiche les éléments de la liste ***L*** dans l'ordre inverse.

Q10) Réécrire la fonction ***Longueur*** et la procédure ***InsererQueue*** sous forme récursive.

Q11) On suppose, cette fois-ci, que les éléments de la liste sont triés. Écrire une procédure ***InsererListeTriee(L, N)*** qui permet d'insérer un entier ***N*** de telle sorte que la liste reste toujours triée.

Remarque : Cette liste triée accepte les doublons.

*** Q12)** Écrire une procédure ***InsererPos(L, N, pos)*** qui permet d'insérer l'entier ***N*** dans la liste ***L*** à la position ***pos.*** Si la position ***pos*** n'existe pas, l'insertion n'est pas effectuée. **Exemple** : Si la liste contient 2 éléments, il est possible d'insérer un nouvel élément à la position **1, 2** ou **3**. Cependant, Il est impossible d'insérer un nouvel élément à la position **4**.

*** Q13)** Réécrire les deux procédures des questions **Q11, Q12** sous forme récursive.

Exercice 2 : Consultation

Écrire une fonction (itérative puis récursive) qui permet de :

a. Vérifier si une valeur ***N*** existe dans la liste. Elle retourne **Vrai**, si ***N*** existe, sinon elle retourne **Faux** *(Fonction **Existe(L, N)**)*.
b. Vérifier si une valeur ***N*** existe dans la liste. Elle retourne l'adresse de l'élément contenant la valeur ***N***. Sinon, elle retourne **Nil** *(Fonction **Adresse1(L, N))**)*.
c. (*) Retourner un pointeur sur l'élément qui se trouve à la position ***pos***. Si la position ***pos*** n'existe pas, la fonction retourne **Nil** *(Fonction **Adresse2(L, pos))**)*
d. Calculer le nombre d'occurrence d'un entier ***N*** dans la liste.
e. Chercher le Maximum *(Fonction **Max(L))**)*. On suppose que la liste n'est pas vide.
f. Vérifier si la liste est triée.

Exercice 3 : Mise à jour

Q1) Écrire une procédure ***SupprimerTete(L)*** qui permet de supprimer le premier élément de la liste ***L***, s'il existe.

Q2) Écrire une procédure ***SupprimerListe(L)*** qui permet de supprimer tous les éléments de la liste ***L***.

Q3) Écrire une procédure ***SupprimerValeur(L, N)*** qui permet de supprimer toutes les occurrences d'une valeur ***N***.

*** Q4)** Écrire une procédure ***SupprimerPos(L, pos)*** qui permet de supprimer l'élément qui se trouve à la position ***pos***.

Q5) Écrire une procédure ***Fusion(L1, L2, L3)*** qui permet de fusionner deux listes triées ***L1*** et ***L2*** et créer la liste triée ***L3***.

Q6) Écrire une procédure ***Eclater(L, L1, L2)*** qui permet d'éclater la liste ***L*** en deux listes ***L1*** et ***L2***. La liste ***L1*** contient les nombres impairs et la liste ***L2*** les nombres pairs.

Q7) Écrire une procédure ***Inverser(L)*** qui permet d'inverser une liste ***L***. Le premier élément devient le dernier, le deuxième devient l'avant dernier et ainsi de suite. **Remarque** : Il est interdit de créer de nouveaux éléments.

Exercice 4 : Piles & Files

Q1) Implémenter une File d'entiers en utilisant une liste linéaire chaînée.

*** Q2)** Une file d'attente avec priorité est une file d'attente dans laquelle l'opération de défilement récupère l'élément le plus prioritaire. Définir le modèle et l'implémenter.

*** Q3)** Implémenter une File d'entiers en utilisant un tableau.

Q4) Implémenter une Pile d'entiers en utilisant une liste linéaire chaînée.

*** Q5)** Implémenter une Pile d'entiers en utilisant un tableau.

**** Travail facultatif***