AP1 TD11 – Quelques opérations de base sur les listes chaînées : correction

On considère dans ce TD une liste chaînée d'entiers non triée.

```
TYPE element = STRUCTURE
        val: entier
        suivant : pointeur d'element
FINSTRUCTURE
```

Exercice 1 – La procédure mystère

Que fait la procédure suivante ? Dans quel cas précis affiche-t-elle « Aïe ouille » ?

```
PROCEDURE mystere(L)
PARAMETRE par VALEUR : L : pointeur d'element
DEBUT
        SI L <> nil ALORS
                TANTQUE L <> nil FAIRE
                        afficher L↑.val
                        L ← L↑.suivant
                FINTANTQUE
        SINON
                afficher « Aïe ouille »
        FINSI
FIN
```

*Cette procédure accepte en paramètre un pointeur de tête **L** et affiche les entiers contenus dans la liste chaînée.*

*Elle affiche « Aie ouille » dans le cas où **L** vaut **nil**, c'est à dire en cas de **liste vide**.*

Exercice 2 – Nombre d'éléments d'une liste

Ecrire un sous programme qui prend en paramètre une liste chaînée d'entiers et qui retourne au programme appelant le nombre d'éléments de cette liste.

```
FONCTION NbElements(L) : entier
PARAMETRE :  L : pointeur d'element
VARIABLE :       nb : entier
DEBUT
        nb← 0
        TANTQUE L <> nil FAIRE
                nb ← nb + 1
                L ← L↑.suivant
        FINTANTQUE
        retourner nb
FIN
```

Exercice 3 – Insertion d'un élément en début de liste

Ecrire un sous programme qui prend en paramètre un entier et qui insère cette valeur en début de liste.

Indication. Faire un dessin !!!

Il est à noter le passage par adresse car le pointeur de tête est modifié par le sous programme.

```
PROCEDURE AjoutDebut(tete, valeur)
PARAMETRE par ADRESSE :   tete : pointeur d'element     { adresse du 1er élément
                                                          de la liste }
PARAMETRE par VALEUR :    valeur : entier               { valeur à insérer }

VARIABLES : ptr : pointeur d'element

DEBUT
        nouvreau(ptr)           { réservation mémoire d'une nouvelle cellule }
        ptr↑.val ← valeur
        ptr↑.suivant ← tete     { l'ancienne tête devient deuxième élément }
        tete ← ptr              { la nouvelle tête est la nouvelle cellule }
FIN
```

Exercice 4 – Insertion d'un élément en fin de liste

Ecrire un sous programme qui prend en paramètre un entier et qui insère cette valeur en fin de liste.

Il est à noter le passage par adresse car le pointeur de tête est modifié par le sous programme en cas de liste vide.

```
PROCEDURE AjoutFin(tete, valeur)
PARAMETRE par ADRESSE :   tete : pointeur d'element     { adresse du 1er élément
                                                          de la liste }
PARAMETRE par VALEUR :    valeur : entier               { valeur à insérer }

VARIABLES : ptr, nouv : pointeur d'element

DEBUT
    SI tete <> nil                  { cas général : liste non vide }
        TANTQUE ptr↑.suivant <> nil FAIRE    { on se place sur le dernier élément }
            ptr ← ptr↑.suivant
        FINTANTQUE
        nouveau(nouv)               { réservation mémoire d'une nouvelle cellule }
        nouv↑.val ← valeur
        nouv↑.suivant ← nil         { ce nouvel élément est le dernier }
        ptr↑.suivant ← nouv         { jonction avec le restant de la liste }
    SINON                           { cas liste vide }
        nouveau(tete)
        tete↑.val ← valeur
        tete↑.suivant ← nil
    FINSI
FIN
```

Exercice 5 – Suppression d'un élément en tête de liste

Ecrire un sous programme qui prend en paramètre une liste chaînée d'entiers et qui supprime le premier élément de cette liste.

Il est à noter le passage par adresse car le pointeur de tête est modifié par le sous programme.

```
PROCEDURE Suppression(tete)
PARAMETRE par ADRESSE :   tete : pointeur d'element     { adresse du 1er élément
                                                          de la liste }
VARIABLES : sauv: pointeur d'element

DEBUT
    SI tete <> nil ALORS            { si la liste n'est pas vide }
        sauv ← tete↑.suivant        { on sauvegarde l'adresse du deuxième élément }
        libere(tete)                { on restitue la zone mémoire dont l'adresse est située
                                      dans tete }
        tete ← sauv                 { l'ancien deuxième élément devient la nouvelle tête }
    FINSI
FIN
```

Exercice 6 – Apothéose

Pour conclure en beauté ces TD d'AP1, on demande de rédiger une version récursive de la procédure « mystère » de l'exercice 1.

Cas général : il s'agit d'exprimer le problème au rang n (afficher toute la liste) par rapport au rang n-1 (afficher toute la liste sauf le premier élément)

Cas d'arrêt : si la liste est vide, on n'affiche rien.

```
PROCEDURE AfficheRecur(tete)
PARAMETRE par VALEUR :    tete : pointeur d'element
DEBUT
    SI tete <> nil ALORS
        afficher tete↑.val
        AfficheRecur(tete↑.suivant)
    FINSI
FIN
```