Chapitre 8 : Partie II – Les listes chaînées

Contenu de cette partie du chapitre :

1. Définition d'une liste chaînées.
2. Gestion dynamique de la mémoire.
3. Listes chaînées simples.
4. Exemples de listes chaînées simples.

1. Définition d'une liste linéaire chaînées

Une ***Liste Linéaire Chaînées (LLC)*** est un ensemble d'éléments d'informations appelés cellules. En plus des informations dont elle est porteuse, une cellule possède un pointeur. Ce pointeur contient l'adresse de la cellule suivante dans la liste.

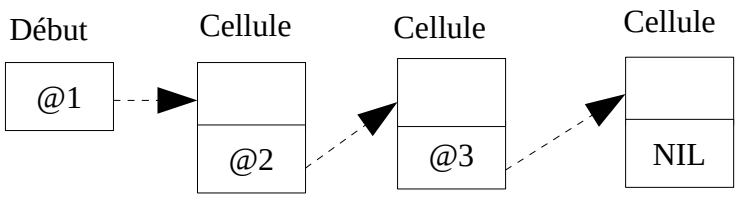

Figure 1 : Liste chaînée de 3 éléments (cellules).

Le dernier élément de la liste chaînée n'a pas de suivant. Son pointeur est donc mis à NIL.

Syntaxe de déclaration :

TYPE pointeur = ^ cellule ;

cellule = **ENREGISTREMENT**

Champs 1 : Type 1;

Champs 2 : Type 2;

…

Champs N : Type N;

Suivant : pointeur;

FIN;

Chaque variable cellule contient :

- Une partie information, représentée par les membres de l'enregistrement, et
- Un pointeur, appelé par exemple "Suivant", contenant l'adresse de la cellule suivante de la liste chaînée.

Exemple :

TYPE ptr_vehicule = ^ véhicule;

véhicule = **ENREGISTREMENT**

No_série : Entier;

Prix : Réel;

Suivant : ptr_vehicule;

FIN;

« No_série » et **« Prix »** représentent les informations dont une cellule est porteuse, et **« Suivant »** est un pointeur qui contient l'adresse de la cellule suivante dans la liste.

2. Gestion dynamique de la mémoire

Une liste chaînée n'est pas limitée à un nombre fixe d'éléments. Elle peut être facilement étendue ou diminuée selon les besoins. C'est la gestion dynamique de la mémoire. Ceci est possible grâce aux fonctions ALLOUER () et LIBÉRER ().

La gestion dynamique de la mémoire permet d'utiliser uniquement l'espace mémoire nécessaire à l'exécution d'un programme. Ce qui n'est pas le cas, lorsqu'on utilise par exemple, un tableau d'enregistrement, car il doit être déclaré avec une dimension fixe.

Rappel :

La fonction ALLOUER () alloue un espace mémoire de la taille de l'enregistrement (cellule) en octets, et fournit l'adresse de cet emplacement en retour. Cette adresse est assignée au pointeur de même type spécifié entre parenthèses.

La fonction LIBÉRER () permet de libérer un emplacement alloué à une cellule lorsqu'on en a plus besoin de cette cellule.

3. Les listes chaînées simples

Une fois une liste chaînée créée, on peut la consulter bien sûr, mais aussi insérer et supprimer des éléments sans restrictions. Ce qui est l'intérêt premier des listes chaînées.

3.1 Création d'une liste chaînée

On donne dans l'algorithme **"Création_liste"** un exemple qui permet de créer une liste chaînée simple à base du type **"ptr_vehicule"** décrit précédemment.

```
ALGORITHME Création_liste;
TYPE ptr_véhicule = ^ véhicule ;
        véhicule = ENREGISTREMENT
                No_série : Entier ;
                Prix : Réel ;
                Suivant : ptr_véhicule ;
        FIN ;
VAR Tête, Courant : ptr_véhicule ;
    Choix : Entier ;
DÉBUT
 ALLOUER (Tête) ;
 Courant ← Tête ;
 RÉPÉTER
   ÉCRIRE ('Entrez le numéro de série du véhicule :') ;
   LIRE (Courant ^.No_série) ;
   ÉCRIRE ('Entrez le prix du véhicule :') ;
   LIRE (Courant ^.Prix) ;
   ÉCRIRE ('Autre saisie ? 1. OUI 2.NON ') ;
   LIRE (Choix) ;
   SI (Choix = 1 ) ALORS
    ALLOUER (Courant ^.Suivant) ;
```

```
    Courant ←     Courant ^ .Suivant ;
  SINON
    Courant ^ .Suivant ← NIL ;
    Courant ← NIL ;
  FIN SI ;
JUSQU'À (Choix = 2)  ;
FIN.
```

3.2 Consultation d'une liste chaînée

Pour consulter la liste créée dans l'algorithme **"Création_liste"**, on utilise le traitement suivant :

```
VAR Tête, Courant : ptr_véhicule ;
DÉBUT
  Courant ← Tête ;
  TANT QUE (Courant < > NIL) FAIRE
    ÉCRIRE (Courant ^.No_série,' ',Courant ^.Prix) ;
    Courant ← Courant ^.Suivant ;
  FIN TANT QUE ;
FIN.
```

3.3 Insertion d'une cellule dans une LLC

Pour insérer un élément dans une LLC, trois cas se présentent :

- On insère la cellule à la tête (au début) de la LLC.
- On insère la cellule au milieu de la LLC.
- On insère la cellule à la queue (à la fin) de la LLC.

3.3.1 Insertion à la tête de la LLC

```
VAR Tête, Nouveau : ptr_véhicule ;
DÉBUT
{On suppose que la liste est créée avec l'algorithme "Création_liste"}
  ALLOUER (Nouveau) ;
  ÉCRIRE ('Entrez le numéro de série du véhicule :') ;
  LIRE (Nouveau ^.No_série) ;
  ÉCRIRE ('Entrez le prix du véhicule :') ;
  LIRE (Nouveau ^.Prix) ;
  Nouveau ^ .Suivant ←    Tête ;
  Tête  ← Nouveau ;
  Nouveau ←NIL ;
FIN.
```

3.3.2 Insertion au milieu de la LLC

Soit **« Courant »** la cellule après laquelle on insère la nouvelle cellule.

```
VAR
      Tête, Nouveau, Courant : ptr_véhicule ;
      X : Entier ;
      Trouve : Booléen ;

DÉBUT
{On suppose que la liste est créée avec l'algorithme "Création_liste"}
  ALLOUER (Nouveau) ;
  ÉCRIRE ('Entrez le numéro de série du véhicule :') ;
  LIRE (Nouveau ^.No_série) ;
```

ÉCRIRE (‘Entrez le prix du véhicule :’) ;
LIRE (Nouveau ^.Prix) ;
Nouveau ^ .Suivant ← NIL ;
ÉCRIRE (‘Donnez le numéro de série du véhicule qui précède le nouveau véhicule à insérer’) ;
LIRE (X) ;
Courant ←Tête ;
Trouve ← Faut ;
TANT QUE (Courant < > NIL) et (Trouve = Faux) **FAIRE**
SI (Courant ^.No_série = X) **alors**
Trouve ←Vrai ;
SINON
Courant ← Courant ^ .Suivant;
FIN SI ;
FIN TANT QUE ;
SI (Courant = NIL) **ALORS**
ÉCRIRE (‘Le véhicule d’avant n’existe pas’) ;
SINON
Nouveau ^ .Suivant ← Courant ^ .Suivant ;
Courant ^ .Suivant ← Nouveau ;
Nouveau ← NIL;
Courant ← NIL ;
FIN SI ;
FIN.

3.3.3 Insertion à la queue de la LLC

VAR Tête, Nouveau, Courant : ptr_véhicule ;

DÉBUT
{On suppose que la liste est créée par l’algorithme “Création_liste”}
ALLOUER (Nouveau) ;
ÉCRIRE (‘Entrez le numéro de série du véhicule :’) ;
LIRE (Nouveau ^.No_série) ;
ÉCRIRE (‘Entrez le prix du véhicule :’) ;
LIRE (Nouveau ^.Prix) ;
Nouveau ^ .Suivant ← NIL ;
Courant ← Tête;
TANT QUE (Courant ^.Suivant < > NIL) **FAIRE**
Courant ← Courant ^ .Suivant);
FIN TANT QUE ;
Courant ^ .Suivant ← Nouveau ;
Nouveau ← NIL;
Courant ← NIL ;
FIN.

3.4 Suppression d’une cellule d’une LCC

Dans la suppression, on peut avoir trois cas aussi :

- On supprime la cellule qui est en tête (au début) de la LLC.
- On supprime une cellule au milieu de la LLC.
- On supprime la cellule qui est en queue (à la fin) de la LLC.

3.4.1 Suppression de la tête de la LLC

On a besoins d'une variable supplémentaire **« Temp »** de type pointeur.

VAR Tête, Temp : ptr_véhicule ;

DÉBUT

{On suppose que la liste est créée par l'algorithme "Création_liste"}

Temp ← Tête ; *{Temp pointera sur la cellule à supprimer}*

Tête ← Tête ^.Suivant ;

LIBÉRER(Temp) ;

Temps ← NIL ;

FIN.

3.4.2 Suppression au milieu de la LLC

On a besoins de deux variables supplémentaires **« Temp »** et **« Courant »** de type pointeur.

VAR

Tête, Temp, Courant : ptr_véhicule ;

X : Entier ;

Trouve : Booléen ;

DÉBUT

{On suppose que la liste est créée par l'algorithme "Création_liste"}

Temp ← Tête ; *{Temp pointe vers la cellule à supprimer}*

ÉCRIRE ('Donnez le numéro de série du véhicule à supprimer') ;

LIRE (X) ;

Trouve ← Faux ;

TANT QUE (Temp < > NIL) et (Trouve = Faux) **FAIRE**

SI (Temp ^.No_série = X) **alors**

Trouve ← Vrai ;

SINON

Temp ← Temp^ .Suivant ;

FIN SI ;

FIN TANT QUE ;

SI (Temp = NIL) **ALORS**

ÉCRIRE ('Le véhicule n'existe pas') ;

SINON

Courant ← Tête ; *{Courant pointera sur la cellule précédente de la cellule à supprimer}*

TANT QUE (Courant ^.Suivant < > Temp) **FAIRE**

Courant ←Courant ^.Suivant ;

FIN TANT QUE ;

Courant ^.Suivant ← Temp ^.Suivant;

LIBÉRER(Temp) ;

Temp ← NIL ;

Courant ← NIL ;

FIN SI ;

FIN.

3.4.3 Suppression de la queue de la LLC

VAR Tête, Temp, Courant : ptr_véhicule ;

DÉBUT

{On suppose que la liste est créée par l'algorithme "Création_liste"}

Temp ← Tête ; *{Temp pointera sur la cellule à supprimer}*

TANT QUE (Temp ^.Suivant < > NIL) **FAIRE**

Temp ← Temp ^.Suivant ;

FIN TANT QUE ;

Courant ← Tête ; *{Courant pointera sur la cellule précédente de la cellule à Supprimer}*

TANT QUE (Courant ^.Suivant < > Temp) **FAIRE**

Courant ←Courant ^.Suivant ;

FIN TANT QUE ;

Courant ^.Suivant ← NIL ;

LIBÉRER(Temp) ;

Temp ← NIL ;

Courant ← NIL ;

FIN.

3.4.4 Suppression de toutes les cellules de la LLC

La suppression d'une liste complète passe par la suppression de toutes ses cellules, **dans le bon ordre**. À l'issue de cette opération, la liste doit être vide et l'ensemble de l'espace mémoire qu'elle occupait libéré.

4. Exemples de listes

Nous avons abordé la liste simplement chaînée qui est la structure la plus simple. Mais bien d'autres existent dont certaines sont très utilisées, en voilà quelques exemples :

• Liste doublement chaînée (bidirectionnelle).

• Pile (LIFO) : Last In first out.

• File (FIFO) : First in First out.

• Anneau (ou liste circulaire).

• Arbre de calcul.

L'ensemble de ces structures de données restent basées sur la notion de pointeur faisant le lien entre différents enregistrements.