ALGORITMIQUE

4.5. LES LISTES CHAÎNEES

4.5.1. Définition

Une liste chaînée est un enchaînement de cellules où, chaque cellule contient deux champs. Le premier champ contient un élément de la liste, et le second champ donne l'adresse de la cellule suivante, sauf la dernière cellule de la liste qui contient l'adresse Nil. Le premier élément de la liste est dit tête, et le dernier élément de la liste est dit queue.

Tout comme les tableaux, les éléments d'une liste chaînée sont de même type.

Contrairement aux tableaux, la taille d'une liste chaînée n'est pas à spécifier lors de la création. Elle peut augmenter en taille indéfiniment.

Toutefois, il convient de noter que certains langages fournissent des tableaux dynamiques, c.-à-d., des tableaux où la taille peut être modifiée en cours d'utilisation.

Contrairement aux tableaux, comme les éléments d'une liste chaînée n'occupent pas forcément des espaces contiguës, il n'est alors pas possible d'accéder directement à un élément quelconque de la liste. L'accès se fait séquentiellement, c.-à-d., passant de cellule en cellule, suivant leur enchaînement.

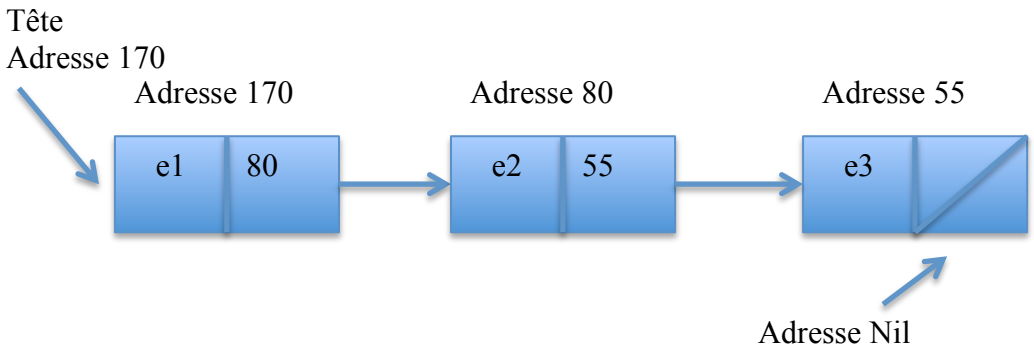
Une liste chaînée peut être définie récursivement comme suit. Une liste chaînée est

- ✓ Soit la liste vide [] ; {condition d'arrêt}
- ✓ Soit la juxtaposition d'une première cellule avec une autre liste (liste qui contient une cellule de moins) {relation de récurrence}.

Ainsi, une liste chaînée est une structure de données récursive.

4.5.2. Représentation

Une liste chaînée est représentée par un pointeur Tête contenant l'adresse de la première cellule, et les cellules de la liste chaînée sont reliées entre elles par des pointeurs.



Ainsi, une liste chaînée est représentée par un pointeur Tête contenant l'adresse de la première cellule qui, à son tour, contient l'adresse de la cellule qui la suit et ainsi de suite jusqu'à la dernière cellule de la liste qui contient l'adresse Nil.

La représentation d'une liste chaînée se fait comme suit.

```
type Liste = ^cellule

cellule = enregistrement

    valeur : t

    suivant : liste

Fin ;
```

4.5.3. Allocation et libération d'un espace mémoire

Quand un pointeur (et donc, une liste chaînée) est déclaré et non initialisé par l'adresse d'une variable existante, on ne peut alors lui réserver par avance un espace mémoire. Il faut lui réserver dynamiquement (en cours d'exécution) un espace mémoire, espace qui, si besoin est, doit être libéré en fin d'utilisation.

Pour allouer dynamiquement un espace mémoire à la cellule sur laquelle pointe p, on utilise la méthode allouer(P).

Pour libérer l'espace mémoire occupé par la cellule sur laquelle pointe P, on utilise la méthode liberer(P).

Exemple

var P : ^entier {on déclare P comme étant un pointeur vers une variable de type entier}

allouer(P) {on alloue à la cellule sur laquelle pointe P un espace mémoire.}

liberer(P) {on libère l'espace mémoire occupé par la cellule sur laquelle pointe P}

4.5.4. Algorithmes

Procédure initialiser (dr L : liste) ;
{On initialise la liste L à nil}
Début
    L := nil
Fin ;

Fonction listevide (d L : liste) : booléenne;
{On teste si L est la liste vide ou pas}
Début

Fin ;

Procédure afficheordrerec (d L : liste) ;
{On affiche les éléments de la liste L selon l'ordre d'apparition dans L}
Début

Fin ;

Ou encore

Procédure afficheordreiter (dr L : liste) ;
{On affiche les éléments de la liste L selon l'ordre d'apparition dans L}
Début

Fin ;

Procédure afficheordreinverserec (d L : liste) ;
{On affiche les éléments de la liste dans l'ordre inverse d'apparition dans L}
Début

Fin ;

Ou encore

Procédure afficheordreinverseiter (dr L : liste) ;
{On affiche les éléments de la liste dans l'ordre inverse d'apparition dans L}
Début

Fin ;

Procédure insertentete (dr L : liste ; d X: t) ;
{On insert l'élément X en tête de la liste L}

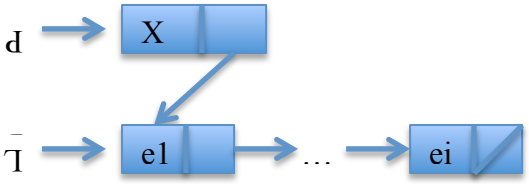
var P : Liste

Début

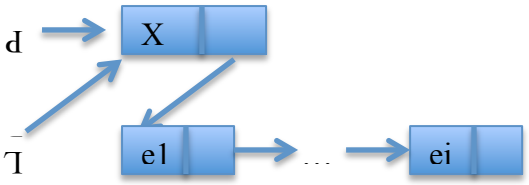
allouer(P) {Comme P non initialisée, alors on réserve à la cellule sur laquelle pointe P un espace mémoire}

P^.valeur := X

P^. suivant := L



L := P



Fin ;

Ainsi, on a

Ou encore

Fonction insertentete (d L : liste ; d X: t) : Liste ;
{On insert l'élément X en tête de la liste L}

var P : Liste

Début

allouer(P)

P^.valeur := X

P^. suivant := L

Insertentete := P

Fin ;

Fonction insertqueue (d L : liste ; d X: t) : Liste ;
{On insert l'élément X en queue de la liste L}

var P, Q: Liste

Début

allouer(P)

P^.valeur := X

P^. suivant := nil {car ce sera la dernière cellule de la liste}

Si (L = nil) alors

Insertqueue := P

Sinon

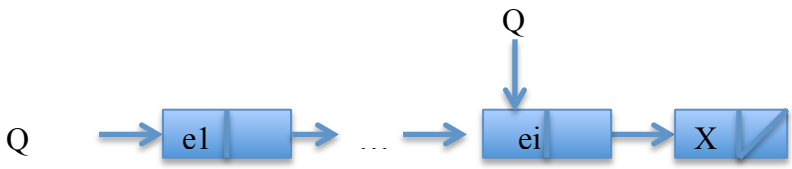
Q := L

Tant que (Q^.suivant ≠ nil) faire

Q := Q^.suivant

Fin tant que {A la sortie du tant que, on atteint la queue de la liste}

Q^.suivant :=P



insertqueue:= Q

Fin ;

Procédure supprentete (dr L : liste) ;
{On supprime l'élément en tête de la liste L}

var P : Liste

Début

Fin ;

Ou encore

Fonction supprentete (d L : liste) : Liste;
{On supprime l'élément en tête de la liste L}

var P : Liste

Début

Fin ;

```
Fonction supprqueue (d L : liste) : Liste;
{On supprime l'élément en tête de la liste L}
var courant, successeur, Ltemp: Liste
Début
```

```
Fin ;
```

Fonction insertion (d L : liste, d X : t) : Liste;
{On insert un élément X dans une liste L triée et on fait en sorte que la liste résultante soit encore triée}

var Ltemp : Liste

Début

Fin ;

Fonction suppression (d L : liste, d X : t) : Liste;
{On insert un élément X dans une liste L triée et on fait en sorte que la liste résultante soit encore triée}

var Ltemp : Liste

Début

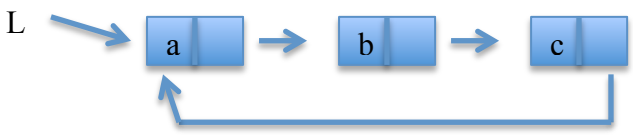
Fin ;

Remarques

On a vu que l'insertion et la suppression des éléments dans un tableau nécessitent un décalage des éléments du tableau, décalage qui a un coût en terme de complexité.

Aussi, si l'on souhaite insérer ou supprimer des éléments, alors il faut utiliser Les listes chainées. Par contre, pour les problèmes de consultation, les tableaux sont plus efficaces.

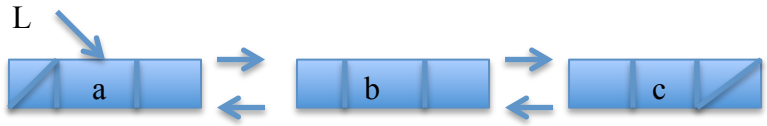
Une liste chainée circulaire est une liste chainée dont la queue pointe sur la tête.



4.5.5. Les listes doublement chaînées

4.5.5.1. Définition

Une liste doublement chaînée est une liste chaînée qui autorise le parcours dans les deux sens. Autrement dit, à partir d'un nœud interne donné, on peut accéder au successeur et au prédécesseur du nœud.



4.5.5.2. Représentation

La représentation d'une liste doublement chaînée se fait comme suit.

```
type ListeDC = ^cellule

cellule = enregistrement

    valeur : t

    precedent, suivant : listeDC

Fin ;
```

4.5.5.3. Algorithmes

```
Fonction insertenteteDC (d L : listeDC ; d X: t) : ListeDC ;
{On insert l'élément  X en tête de la liste L}
var P : ListeDC
Début




Fin ;
```

Fonction supprenteteDC (d L : listeDC) : ListeDC;
{On supprime l'élément en tête de la liste L}

var P : Liste

Début

Fin ;