LES LISTES CHAINÉES

DÉFINITION

- Une liste chaînée désigne une structure de données représentant une collection ordonnée et de taille arbitraire d'éléments de même type.
- L'accès aux éléments d'une liste se fait de manière séquentielle : chaque élément permet l'accès au suivant (contrairement au cas du tableau dans lequel l'accès se fait de manière absolue, par adressage direct de chaque cellule dudit tableau).

PRINCIPE DE FONCTIONNEMENT

- Le principe de la liste chaînée est que chaque élément possède, en plus de la donnée, des pointeurs vers les éléments qui lui sont logiquement adjacents dans la liste.
- L'usage d'une liste est bien plus rapide que celui d'un tableau pour les insertions et les suppressions d'éléments.
- L'usage d'un tableau (direct) est bien plus rapide que celui d'une liste (séquentiel) pour l'accès aux éléments.

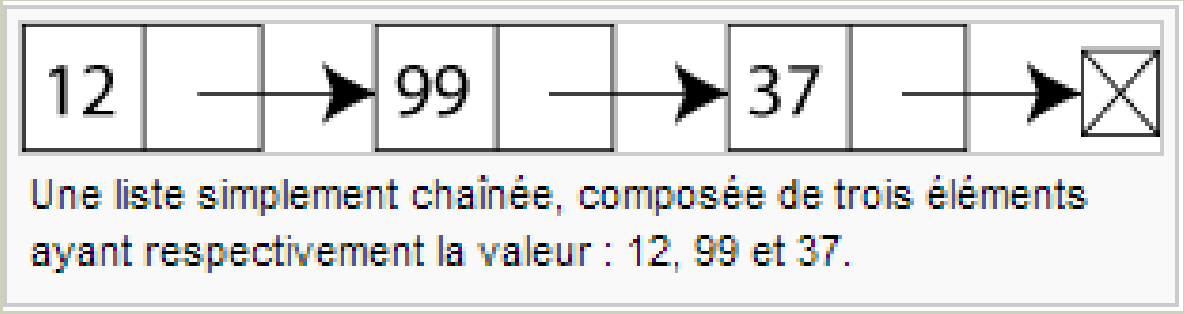
TYPES DE LISTES CHAÎNÉES

On distingue trois types majeurs de listes chainées:

- La liste simplement chainée
- La liste doublement chainée
- le cycle

LISTE SIMPLEMENT CHAINÉE

- Comme on le voit sur le schéma ci-contre, deux informations composent chaque élément de la liste chaînée :
  - la valeur associée à l'élément (ou donnée d'exploitation).
  - un pointeur vers l'élément suivant (ou successeur).
  - On parle aussi de tête de liste et de queue de liste.



Une liste simplement chaînée, composée de trois éléments ayant respectivement la valeur : 12, 99 et 37.

- Comme un seul élément de la liste est pointé, l'accès se fait uniquement dans un sens.

REPRÉSENTATION

Enregistrement Elément
- Donnée_1: Type_1
- Donnée_2: Type_2
- …
- *suivant : Elément

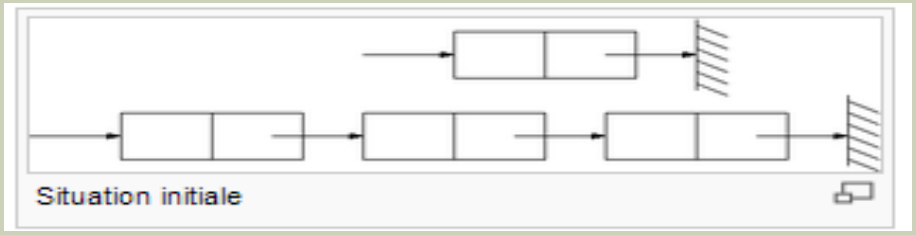
Fin_Enregistrement

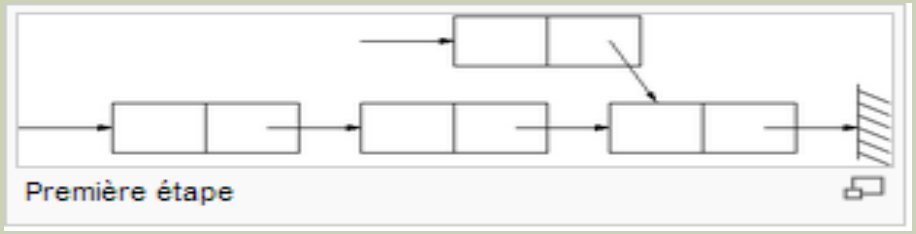
TYPE Liste = *Elément

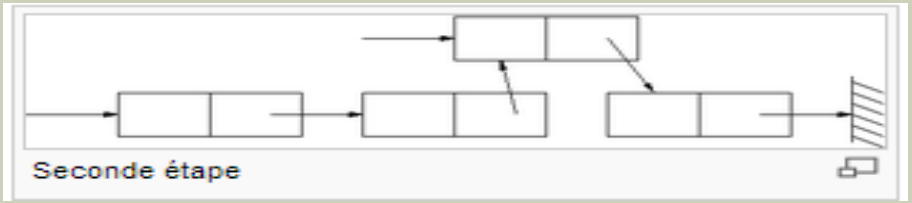
La liste vide est représentée par le ***pointeur NULL***.

AJOUT D'UN ÉLÉMENT

- Voici la représentation schématique de l'ajout d'un nouvel élément *f* après un élément *e* se trouvant dans une liste simplement chaînée. La procédure se fait en deux étapes :
  - le successeur de *e* devient le successeur de *f* ;
  - *f* devient le successeur de *e*.

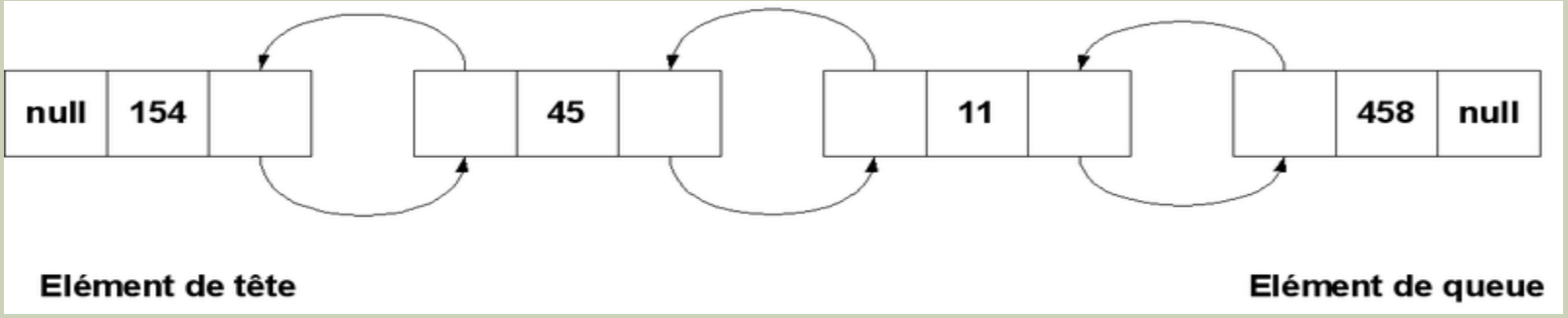






LISTE DOUBLEMENT CHAINÉE

- La différence avec une liste simplement chaînée est que, cette fois-ci, un pointeur vers l'élément précédent (ou prédécesseur) est ajouté. Ainsi l'accès peut se faire indifféremment dans les deux sens : de successeur en successeur, ou de prédécesseur en prédécesseur.
- Cette structure est plus coûteuse en mémoire et en nombre d'instructions pour la mise à jour. Par contre, cela permet d'opérer une insertion avant ou après un élément, sans devoir disposer d'un pointeur sur un voisin, là ou une liste simplement chaînée n'autorise une insertion qu'à une seule position par rapport à un élément



REPRÉSENTATION

Enregistrement Elément

- Donnée_1: Type_1
- Donnée_2: Type_2
- …
- *précédent: Elément
- *suivant : Elément

Fin_Enregistrement

TYPE Liste_bidirectionnelle = *Elément

CYCLE

- Le cycle est la propriété que présente une liste chaînée de former une boucle (ou chaîne fermée). La notion de début de chaîne ou de fin de chaîne disparaît.
- Une liste cyclique (ou circulaire) est créée lorsque le dernier élément possède une référence vers le premier élément (si la liste est doublement chaînée, alors le premier élément possède aussi une référence vers le dernier).
- L'utilisation de ce type de liste requiert des précautions pour éviter des parcours infinis, par exemple, lors d'une recherche vaine d'éléments.