Les piles

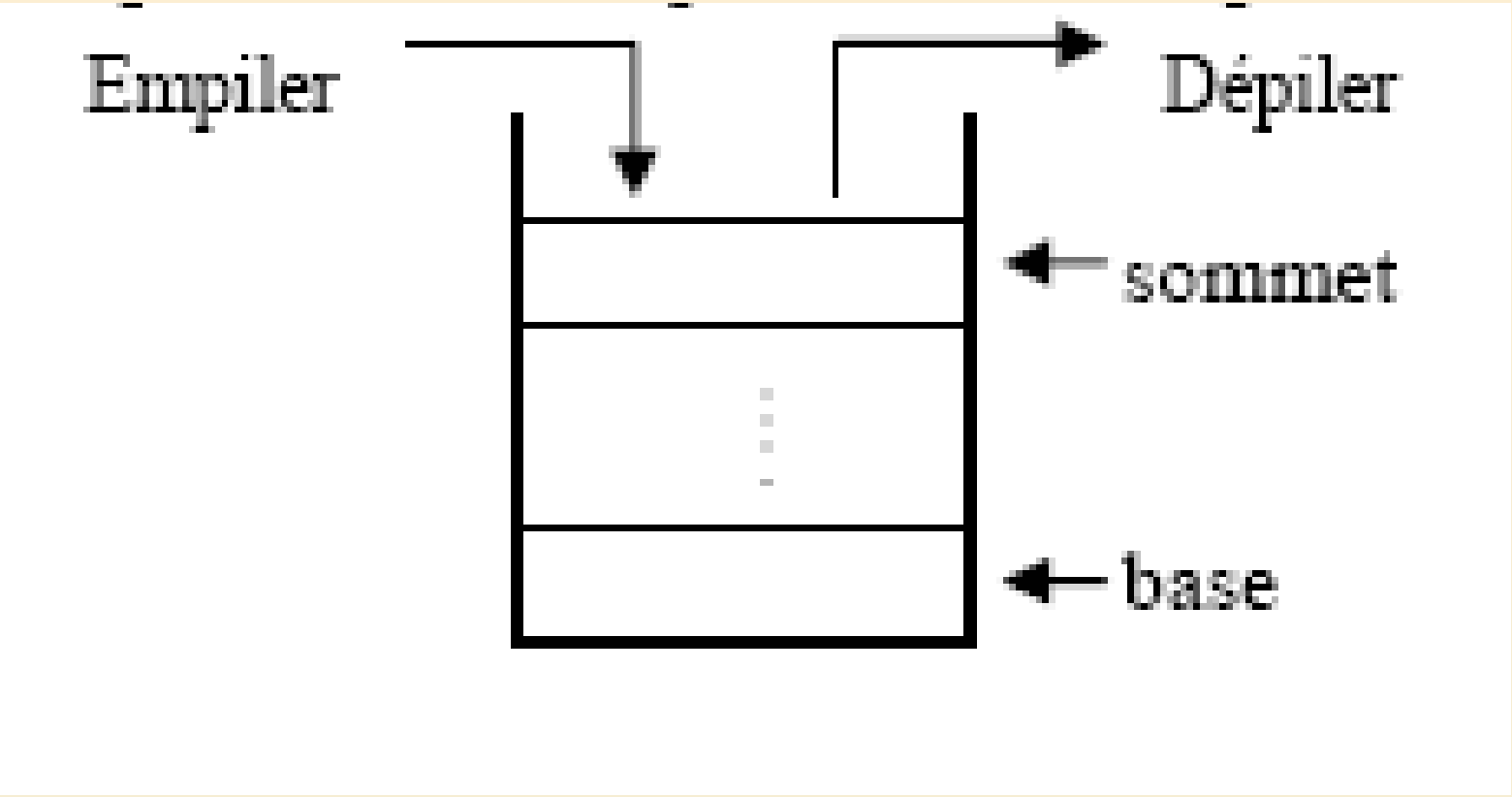
Une pile est une liste chaînée d'informations dans laquelle :

- Un élément ne peut être ajouté qu'au sommet de la pile,
- Un élément ne peut être retiré que du sommet de la pile.

Il s'agit donc d'une structure de type LIFO (Last In First Out). On ne travaille que sur le sommet de la pile.

Les piles sont comparables à des piles d'assiettes.

Les piles



Les piles

On associe à une pile les termes de :

- PUSH pour empiler c'est-à-dire ajouter un élément,
- POP pour dépiler c'est-à-dire supprimer un élément.

Les piles servent à revenir à l'état précédent et sont utilisées pour :

- implanter les appels de procédures (pour revenir à l'état d'avant l'appel),
- annuler une commande,
- évaluer des expressions arithmétiques,
- etc…

Les piles

Opérations autorisées

Les opérations autorisées avec une pile sont :

- empiler, toujours au sommet, et jusqu'à la limite de la mémoire,
- dépiler, toujours au sommet, si la pile n'est pas vide,
- vérifier si la pile est vide ou non.

Les piles

On peut implémenter une pile dans un tableau (pile statique) ou dans une liste chaînée (pile dynamique).

C'est l'implémentation en liste chaînée qui est présentée ici. Le sommet de la pile est le premier élément et le pointeur de tête pointe sur ce sommet. Il faut commencer par définir un type de variable pour chaque élément de la pile. La déclaration est identique à celle d'une liste chaînée,

par exemple pour une pile de chaînes de caractères :

Les piles

Exemple:

```
Type Pile = ^Element
        Element = Structure
        Info : chaîne de caractères;
        Suivant : Pile
Fin.
```

Les piles

Empiler

Empiler un élément revient à faire une insertion en tête dans la liste chaînée.

Exemple:

```
Procedure Empiler (Tête : Pile, Valeur : chaîne de caractères)
// Ajout d'un élément dans une pile passée en paramètre
Variable  P : Pile;                    // pointeur auxiliaire
```

Les piles

DEBUT

Allouer(P) // Réserve un espace mémoire pour le nouvel élément

P^.Info := Valeur //stocke dans l'Info de l'élément pointé par P la valeur passée en paramètre

P^.Suivant := Tête //stocke dans l'adresse du Suivant l'adresse de Tête

Tête := P // Tête pointe maintenant sur le nouvel élément

FIN.

Les piles

Dépiler

Dépiler revient à faire une suppression en tête.

Procedure Dépiler (Tête : Pile)

// Suppression de l'élément au sommet de la pile passée en paramètre

//Pointeur nécessaire pour libérer la place occupée par l'élément dépilé

Variable P : Pile;

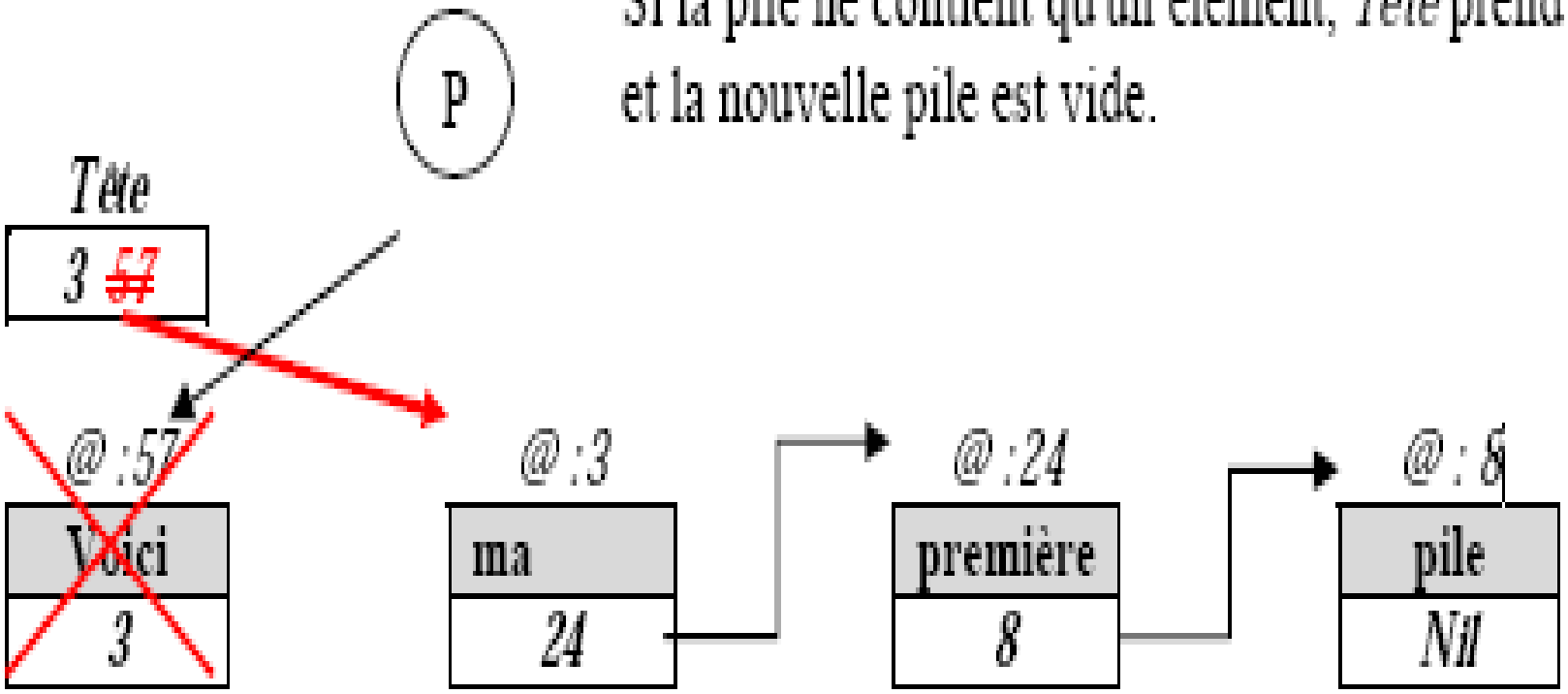
DEBUT

// Vérifier si la pile est vide

```
        SI Tête <> NIL ALORS
                P :=  Tête;
                Tête :=  Tête^.Suivant;
                Désallouer(P);
        FINSI
FIN
```

Les piles

Si la pile ne contient qu'un élément, *Tête* prend la valeur *Nil* et la nouvelle pile est vide.



Les files

Une file, ou file d'attente, est une liste chaînée d'informations dans laquelle :

- Un élément ne peut être ajouté qu'à la queue de la file,
- Un élément ne peut être retiré qu'à la tête de la file.

Il s'agit donc d'une structure de type FIFO (First In First Out). Les données sont retirées dans l’ordre où elles ont été ajoutées.

Une file est comparable à une queue de clients à la caisse d'un magasin.

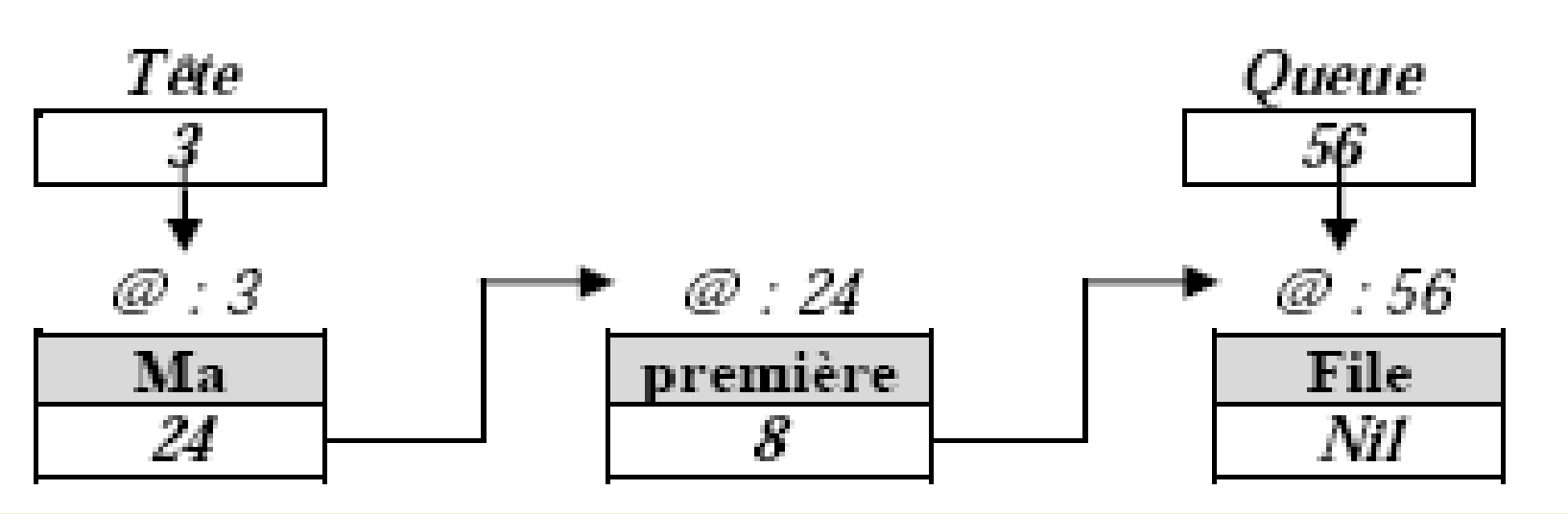
Les files

Les files servent à traiter les données dans l'ordre où on les a reçues et permettent de :

- gérer des processus en attente d'une ressource système (par exemple la liste des travaux à
- éditer sur une imprimante)
- construire des systèmes de réservation
- etc.

Les files

Pour ne pas avoir à parcourir toute la liste au moment d'ajouter un élément en queue, on maintient un pointeur de queue. Attention une file peut très bien être simplement chaînée même s'il y a un pointeur de queue.



Les files

Opérations autorisées

Les opérations autorisées avec une file sont :

- enfiler toujours à la queue et jusqu'à la limite de la mémoire,
- défiler toujours à la tête si la file n'est pas vide,
- vérifier si la file est vide ou non.

Les files

On peut implémenter une file dans un tableau (file statique) ou dans une liste chaînée (file dynamique). C'est l'implémentation en liste chaînée qui est présentée ici.

Le pointeur de tête pointe sur le premier élément de la file, et le pointeur de queue sur le dernier. Il faut commencer par définir un type de variable pour chaque élément de la file.

La déclaration est identique à celle d'une liste chaînée, par exemple pour une file de chaînes de caractères :

Les files

```
Type File = ^Element
     Element = Structure
          Info : chaîne de caractères;
          Suivant : File
Fin;
```

Les files

Enfiler

Enfiler un élément consiste à l'ajouter en queue de liste. Il faut envisager le cas particulier où la file était vide. En effet, dans ce cas, le pointeur de tête doit être modifié.

Les files

```
Procedure  Enfiler (Tête, Queue : File, Valeur : variant)
//  Ajout d'un élément dans une file
Variable  P : File ;

DEBUT
        Allouer(P) ;
        P^.Info :=   Valeur ;
        P^.Suivant  :=  Nil ;
                SI Tête = Nil ALORS
                                Tête :=  P
                        SINON
                                Queue^.Suivant  :=  P;

        Queue  :=  P
FIN;
```

Les files

Défiler

Défiler est équivalent à dépiler et consiste à supprimer l'élément de tête si la file n'est pas vide. Si la file a un seul élément, il faut mettre à jour le pointeur de queue car on vide la file.

Il faut conserver l'adresse de l'élément qu'on supprime pour libérer sa place.

Les files

Procedure Défiler (*Tête, Queue : File, Valeur : chaîne de caractères)*

```
// Suppression de l'élément de tête de la file passée en paramètre
Variable P : File ;

DEBUT
        SI Tête <> NIL ALORS
                Valeur :=  Tête^.info ;
                P :=  Tête ;
                Tête :=  Tête^.Suivant ;
                Désallouer(P) ;

                Si Tête = Nil ALORS
                        Queue :=  Nil ;
                FINSI
        FINSI
FIN;
```