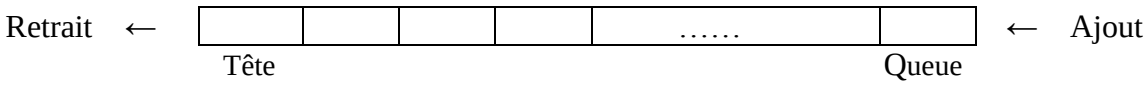
Chapitre 5 : Les Files

Définition : une file est une collection d'éléments de même type où les insertions des nouveaux éléments se font toutes à la fin (queue) et les suppressions toutes au début (tête de file). Tout ajout d'élément se fait par la *queue* (après le dernier) et tout retrait se fait par la *tête* (on retire le premier élément de la file). L'accès à un élément quelconque se fait après le retrait (défilement) de tous les éléments qui le précèdent.

Une file est une structure de donnée de type ***FIFO***
(« First In, First Out » = « premier arrivé, premier sorti »).

Exemples : file d'attente, …

Schématiquement, une file est représentée comme suit :



I.2 Les Opérations sur les Files :

Les opérations sur les files sont les suivantes :

- ***Ajout*** d'un élément : l'action consistant à ajouter un nouvel élément et le mettre au dernier s'appelle ***enfiler***.
- ***Suppression*** d'un élément : l'action consistant à retirer le premier élément (en tête de la file) s'appelle ***défiler***.
- ***Consultation*** : consulter le premier élément de la file.

I.3. Utilisation des Files dans les Applications Informatiques

Les files sont des structures de données très utilisées aussi bien dans la vie courante que dans les systèmes informatiques. Parmi leurs domaines d'application :

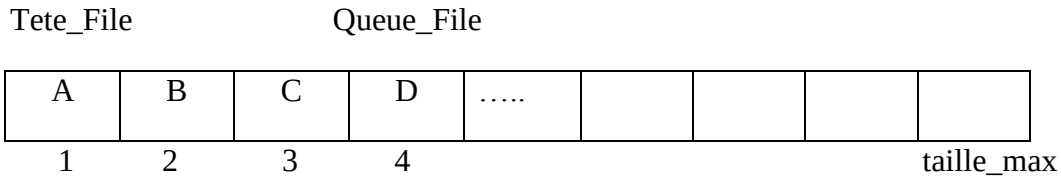
- Les programmes de traitement de transactions telles que les réservations de sièges d'avion ou de billets de théâtre.
- Les systèmes d'exploitation pour gérer l'allocation des ressources (les imprimantes par exemple).
- Dans le domaine des télécommunications pour gérer les appels téléphoniques et la délivrance des messages.

II. REPRESENTATION DES FILES EN MEMOIRE :

De même que pour les piles, il existe deux représentations (implémentations) des files en mémoire : contiguë et chaînée.

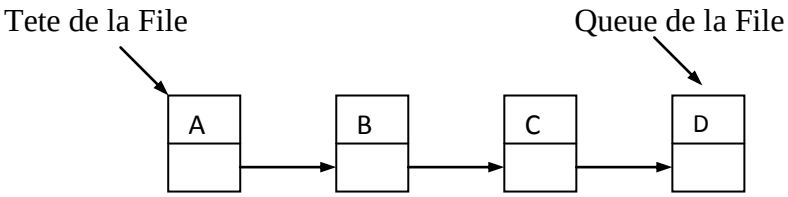
II.1 Représentation d'une File par un Tableau

On peut représenter une file par un tableau (forme contiguë), alloué d'une manière statique ou dynamique.



II.2 Représentation d'une File par une Liste Chainée

Une file peut être représentée par une liste chainée où l'ajout d'un élément se fait toujours par la queue de la liste et le retrait par la tête.



III. PRIMITIVES DE MANIPULATION DES FILES

A l'instar des piles, ces primitives correspondent à des fonctions de base de manipulation des files. Leur écriture dépend de l'implémentation de la file (contigüe ou chainée) et du type des éléments de la file.

InitFile : permet d'initialiser la file

FileVide : permet de vérifier si la file est vide.

FilePleine : permet de vérifier si la file est pleine (cas d'allocation contigüe).

TeteFile : retourne l'élément se trouvant en tête de file dans une variable sans le défiler.

Enfiler : permet d'ajouter un élément en queue de file.

Défiler : permet de retirer (supprimer) l'élément se trouvant en tête de file et retourne sa valeur dans une variable.

.

III.1 Opérations de Manipulation des Files dans les deux Représentations

On considère une file d'éléments de type TypeElement. On écrira les primitives dans les deux types d'implémentation (contigüe et chainée).

A) Représentation Contigüe (Cas d'Allocation Statique)

1) Déclaration

```
constante   max 100
type File = Enregistrement
                T : tableau [max] TypeElement ;
                Tete, Queue : entier ;
            FinEnre
variable f : File ;
```

2) Initialisation

```
Fonction  InitFile ( ) : File
Variable f : File
Debut
   f.Tete ←1;     f.Queue ←0;
   retourner (f);
Fin
```

3) Test si la file est vide

```
Fonction FileVide (E/ f : File) : booleen
 Debut  si (f.Queue < f.Tete) alors retourner (vrai) ;
                              sinon retourner (faux) ;
        fsi
  Fin
```

4) Test si la file est pleine

```
Fonction FilePleine (E/ f : File) : booleen
Debut
  si (f.Queue=max)  alors retourner (vrai) ;
                    sinon retourner (faux) ;
  fsi
Fin
```

5) Consultation de la tete de file

```
Fonction TeteFile (E/ f : File) : TypeElement
variable x : TypeElement ;
Debut
  x←f.T[f.Tete];
  retourner (x) ;
Fin
```

6) Ajout d'un element

```
Procedure Enfiler (E/S f: File, E/ x : TypeElement)
Debut
  f.Queue ←f.Queue+1 ;
  f.T[f.Queue]←x ;
Fin
```

7) Suppression d'un élément

```
Procedure Défiler (E/S f: File, E/S x : TypeElement)
Debut
   x←f.T[f.Tete] ;
   f.Tete ←f.Tete+1 ;
 Fin
```

Remarque : A tout moment le nombre d'éléments dans la file est F.queue- F.tête + 1

B) Représentation Chainée

1) Déclaration

```
type Element = Enregistrement
                  info : TypeElement ;
                  suivant : ^ Element ;
               FinEnreg
```

```
Type File = Enregistrement
                Tete : ^ Element ;
                Queue : ^ Element ;
            FinEnreg
variable f : File ;
```

2) Initialisation

```
Fonction InitFile ( ) :  File
variable   f : File
Debut
  f.Tete←nil ;
  f.Queue←nil ;
  retourner (f) ;
Fin
```

3) Test si la file est vide

```
Fonction FileVide (E/ f: File) : booleen
Debut
  si (f.Tete=nil) alors retourner (vrai)
                  sinon retourner (faux) ;
  fsi
Fin
```

4) Consultation de la tete de file

```
Fonction TeteFile (E/ f : File) : TypeElement
variable   x : TypeElement
 Debut
  x← ^(f.Tete).info;
  retourner (x) ;
Fin
```

5) Ajout d'un element

```
Procedure Enfiler (E/S f : File , E/ x : TypeElement)
variable temp : ^ Element ;
Debut
  temp←Allouer(TailleDe(Element)) ;
  ^temp.inf←x ;
  ^temp.suivant←nil ;
   si (f.Tete=nil) alors   f.Tete←temp ;
                           f.Queue←temp ;
                   sinon  ^(f.Queue).suivant←temp ;
                          f.Queue←temp ;
   fsi
Fin
```

6) Suppression d'un élément

```
Procedure Defiler (E/S f : File , E/S x : TypeElement)
variable temp : ^ Element ;
Debut
  x← ^(f.Tete).info ;
  temp←f.Tete ;
  si (f.Tete=f.Queue)  alors f.Tete ← nil ;
                             f.Queue ← nil ;
                       sinon f.Tete←^(f.Tete).suivant ;
  fsi
 liberer (temp) ;
Fin
```