

# **Chapitre 11 : Les Files d'attente**

## **Introduction**

Une file d'attente est une structure de données complémentaire qui fonctionne avec le principe du FIFO (FIRST IN FIRST OUT). Les files d'attente sont utilisées dans les applications de gestion de réservation et/ou de gestion priorité des processus. Une file d'attente fonctionne essentiellement sur la base de primitives. Les primitives associées sont réalisées soit avec des pointeurs soit avec des tableaux.

## **I - Les primitives associées aux files d'attente**

Ces primitives sont des méthodes qui permettent d'appliquer des opérations spécifiques à une ou des files d'attente. Elles sont :

- **initFile()**
- **fileVide()**
- **filePleine()**
- **Enfiler()**
- **Defiler()**

### **1) La primitive initFile()**

C'est une procédure qui permet d'initialiser les arguments d'une file d'attente passés en paramètres.

### **2) La primitive fileVide()**

C'est une fonction qui reçoit les arguments d'une file d'attente puis retourne **vrai** si la file d'attente est vide et **faux** dans le cas contraire.

### **3) La primitive filePleine()**

C'est une fonction qui reçoit les arguments d'une file d'attente puis retourne **vrai** si la file d'attente est pleine et faux dans le cas contraire

### **4) La primitive Enfiler() ou AjouterFile()**

C'est une procédure qui reçoit les arguments d'une file et une valeur puis ajoute la valeur dans la file tout en respectant le principe du FIFO (Toute valeur est ajoutée en queue de file si la file n'est pas vide sinon elle est placée en Tete de file).

**NB : La primitive Enfiler() ne peut être utilisée que si la file n'est pas pleine.**

### 5) La primitive Defiler() ou EnleverFile()

C'est une procédure qui reçoit les paramètres d'une file d'attente puis extrait la valeur située en Tete de file afin de la sauvegarder dans une variable pour d'éventuels traitements. Toute valeur extraite de la file y est automatique supprimée.

**NB : La file ne peut être défilée que si elle n'est pas vide.**

### Remarque générale :

L'implémentation d'une file d'attente est possible en utilisant soit les pointeurs (sous forme de liste dynamique) soit en utilisant les tableaux (vecteur c'est à dire sous forme de liste contigue)

## II - Implémentation d'une file d'attente en utilisant les pointeurs

Une file d'attente réalisée sous forme de pointeurs a la même structure que la liste monodirectionnelle sauf que la file d'attente a deux (2) voies d'accès qui sont Tete et Queue. Tete contient l'adresse du premier élément et Queue contient l'adresse du dernier élément.

### A - Déclaration de la file d'attente

#### a. Syntaxe :

```
Type nomFileAttente = ↑ Structure
    Debut
        Info(s) : Type(s)
        Suiv : nomFileAttente
    Fin
    Var Tete, Queue : nomFileAttente
```

Exemple 1 : File d'attente d'entiers réalisée sous forme de pointeurs

```
Type FileEntiers = ↑ Structure
    Debut
        Info : entier
        Suiv : FileEntiers
    Fin
    Var Tete, Queue : FileEntiers
```

Exemple 2 : File d'attente de processus réalisée sous forme de pointeur. Processus (id, nom, etat, taille)

**Type PROCESSUS = Structure**

**Début**

**id : entier**

**nom, etat : chaine**

**taille : reel**

**Fin**

**Type FileProcessus = Structure**

**Début**

**info : PROCESSUS**

**suiv : FileProcessus**

**Fin**

**Var T, Q : FileProcessus**

## **B - Réalisation des primitives en utilisant les pointeurs**

Une file d'attente réalisée avec les pointeurs ressemble à une liste monodirectionnelle c'est à dire a la même description que la liste monodirectionnelle. La file d'attente peut disposer de plusieurs champs informations mais elle a exactement un champ pointeur. La file d'attente a 2 voies d'accès qui sont Tete et Queue.

Tete contient l'adresse du premier element de la file

Queue contient l'adresse du dernier element de la file.

La file d'attente a des maillons chainés du premier au dernier

Le traitement des maillons se fait en respectant le principe du FIFO.

### **1- La primitive initFile()**

Pour initialiser une file d'attente réalisée avec les pointeurs, il faut affecter à Tete et à Queue la valeur NIL.

**Exercice d'application :**

Soit une file d'attente d'entiers réalisée avec les pointeurs, écrire un module qui réalise la primitive initFile().

**Solution**

```
Type File = ↑Structure
Debut
    info:entier
    suiv:File
Fin
Var Tete,Queue:File
Procédure initFile(D/R Tete,Queue:File)
Debut
    Tete <- Nil
    Queue <- Nil
Fin
```

**2- La primitive fileVide()**

Elle reçoit les arguments d'une file puis renvoie VARI si Tete = Queue = Nil et FAUX dans le cas contraire.

**NB : Une file qui n'a pas de premier n'a pas non plus de dernier et vice versa.**

**Exercice d'application :**

Soit une file d'attente d'entiers réalisée avec les pointeurs, écrire un module qui réalise la primitive fileVide().

**Solution**

```
Type File =↑Structure  
Debut  
    info:entier  
    suiv:File  
Fin  
Var Tete,Queue:File  
Fonction fileVide(Donnees Tete,Queue:File):booleen  
Debut  
    Si (Tete = Nil et Queue = Nil) Alors  
        retourner vrai  
    Sinon  
        retourner faux  
    FinSi  
Fin
```

### 3- La primitive filePleine()

Une file réalisée avec les pointeurs est pleine si toutes les ressources mémoires sont occupées. Dans ce cas, la primitive renvoie VRAI sinon elle renvoie FAUX.

#### Exercice d'application :

Soit une file d'attente d'entiers réalisée avec les pointeurs, écrire un module qui réalise la primitive filePleine().

#### Solution

**Type File = ↑Structure****Debut****info:entier****suiv:File****Fin****Var Tete,Queue:File****Fonction filePleine(Donnees Tete,Queue:File):booleen****var p:File****Debut****Allouer(p)****Si (p = Nil) Alors****Ecrire "Toutes les ressources memoires sont occupees"****retourner Vrai****Sinon****Liberer(p)****retourner Faux****FinSi****Fin**

#### 4- La primitive **enfiler()** ou **ajouterFile()**

Elle recoit les arguments d'une file et une valeur à ajouter dans la file. La primitive ajoute la valeur dans la file si la file n'est pas pleine. L'ajout se fait toujours en Queue de file si la file n'est pas vide sinon l'élément ajouté devient premier et dernier.

#### **Exercice d'application :**

Soit une file d'attente d'entiers réalisée avec les pointeurs, écrire un module qui reçoit les paramètres de la file et une valeur à ajouter puis réalise la primitive **enfiler()**.

Solution

**Type File = ↑Structure**

**Debut**

**info:entier**

**suiv:File**

**Fin**

**Var Tete,Queue:File**

**Procédure enfiler(Donnee Val:Entier**  
                    **D/R Tete,Queue:File)**

**Var pval:File**

**Debut**

**Si (filePleine(Tete,Queue)=vrai) Alors**

**Ecrire "Impossible d'ajouter car la file est pleine"**

**Sinon**

**Allouer(pval)**

**pval↑.info ← Val**

**pval↑.suiv ← Nil**

**Si (fileVide(Tete,Queue)=Nil) Alors**

**Tete ← pval**

**Queue ← pval**

**Sinon**

**Queue↑.suiv ← pval**

**Queue ← pval**

**FinSi**

**FinSi**

**Fin**

## 5- La primitive defiler() ou enleverFile()

Elle recoit les arguments d'une file et extrait la valeur située en Tete de file pour la sauvegarder dans une variable pour d'éventuels traitements. Toute valeur extraite est automatiquement supprimée de la file. L'extraction ne peut se faire que si la file n'est pas vide.

**Exercice d'application :**

Soit une file d'attente d'entiers réalisée avec les pointeurs, écrire un module qui reçoit les paramètres de la file puis réalise la primitive defiler().

**Solution****Type File =  $\uparrow$ Structure****Debut****info:entier****suiv:File****Fin****Var Tete,Queue:File****Procédure defiler(D/R Tete,Queue:File****Resultat val:entier)****var pval:File****Debut****Si (fileVide(Tete,Queue) = vrai) Alors****Ecrire "Impossible d'extraire car la file est vide"****Sinon****pval  $\leftarrow$  Tete****val  $\leftarrow$  Tete $\uparrow$ .info****Si (Tete = Queue) Alors //File composee d'un element****initFile(Tete,Queue)****Sinon****Tete  $\leftarrow$  Tete $\uparrow$ .suiv****FinSi****Liberer(pval)****FinSi****Fin**



**Exercice d'application 1 :**

Soit un tableau de 150 produits, écrire un module qui transfère dans une file d'attente les produits de catégorie Alimentaire. Produit (code, nom, categorie, prix unitaire, quantite)

**Const N = 150****Type Produit = Structure****Debut****code, nom, categorie:chaîne****prixUnitaire:entier****quantite:reel****Fin****Type Tab = Tableau[1..N]Produit****Type FileProduit = ↑Structure****Debut****info : Produit****suiv : FileProduit****Fin****Var Tete,Queue:FileProduit****T:Tab****Procedure CreationFile(Donnees T:Tab****N:Entier****Resultats Tete,Queue:FileProduit)****var i:entier****Debut****initFile(Tete,Queue)****Pour i ← 1 a N Faire****Si (T[i].categorie = "Alimentaire") Alors****Enfiler(T[i],Tete,Queue)****FinSi****FinPour**

**Fin****Exercice d'application 2 :**

Soit une file d'attente de processus, écrire un module qui transfère dans un tableau les processus qui sont éligibles. Un processus est caractérisé par son id, son nom, son état (élu, éligible, bloqué) et sa taille.

**Const N = 100****Type PROCESSUS = Structure****Debut****id:entier****nom:chaîne****etat: "élu","éligible","bloqué"****taille:reel****Fin****Type TabProcessus = Tableau[1..N] PROCESSUS****Type FileProcessus = ↑Structure****Debut****info:Processus****suiv:FileProcessus****Fin****var T:Tab****Tete,Queue:FileProcessus****Procedure Application(Donnees Tete,Queue:FileProcessus****Resultats T:Tab****N:Entier)****var p: PROCESSUS****i:entier****Debut****i ← 0****TantQue(FileVide(Tete,Queue)=faux) Faire****Defiler(Tete,Queue,p)****Si (p.etat="Eligible") Alors****i ← i + 1****T[i] ← p****FinSi**

**FinTantQue****N  $\leftarrow$  i****Fin****EXERCICE 0 :**

Soit une file d'attente d'entiers, écrire un sous-programme qui détermine le nombre de nombres premiers de la file d'attente.

**Type File =  $\uparrow$ Structure****Debut****info : entier****suiv : File****Fin****Var Tete, Queue:File****Fonction DecompteNombrePremier(Donnees Tete,Queue:File):entier****var val,i,cpt,nbPrem:entier****Debut****nbPrem  $\leftarrow$  0****Tantque(fileVide(Tete,Queue)=Faux)Faire****Defiler(Tete,Queue,val)****cpt  $\leftarrow$  0****Pour i allant de 1 à val Faire****Si (val mod i = 0) Alors****cpt  $\leftarrow$  cpt + 1****FinSi****FinPour****Si (cpt = 2) Alors****nbPrem  $\leftarrow$  nbPrem + 1****FinSi****FinTantQue**

**retourner (nbPrem)**

**Fin**

**EXERCICE 0 bis :**

Soient une file d'attente réalisée sous la forme de pointeurs et une liste monodirectionnelle d'entiers, écrire un sous-programme qui détermine et affiche le nombre de présence de chaque valeur de la file dans la liste.