

Files






1. Définition

- Structure de données à accès restreint
 - Éléments considérés en fonction de leur ordre d'arrivée
 - ❑ Ajout en fin de file
 - ❑ Retrait en début de file
- FIFO : First In First Out

Exemple : Une file d'attente à un guichet, dans un magasin



Files

2. Primitives

- Initialisation de la file
  `init_file(f)`
- Test de l'état de la file
  `file_vide(f)`
- Ajout d'un élément en fin de file
  `ajout(f,v)`
- Retrait de l'élément en début de file
  `retrait (f,v)`
- Renvoi de la valeur en tête de file
  `Consultation_Tete(f)`

Files

3. Implémentation sous forme de tableaux

- Tableau + deux entiers = (tete, queue)
- tete : indice de la prochaine case à vider
- queue : indice de la dernière case remplie
- Retraits  apparition cases vides dans le tableau
- Queue = nbelt  impossible d'ajouter de nouveaux éléments!

Files

3. Implémentation sous forme de tableaux

- Solution 1 : Décalages des éléments d'une case vers le bas à chaque retrait
➡ solution coûteuse à cause des recopies systématiques
- Solution 2 : Amélioration de la solution1
➡ Décalage uniquement quand queue = nbelt
- Solution 3 : Gestion du tableau sous forme de tableau circulaire (anneau)
➡ Plus aucun déplacement
➡ Solution basée sur l'utilisation de l'opérateur modulo

Files

3. Implémentation sous forme de tableaux – Solution 3

- Déclaration

const nbelt = 100

type file = **enregistrement**

 tab = **tableau**[0..nbelt-1] **de type entier**

 tete, queue : **entier**

fin_enregistrement

Files

3. Implémentation sous forme de tableaux

- Initialisation de la file

```
procedure init_file(var f :file)
```

```
/* la file est vide : tete ne désigne aucune case à vider*/
```

```
/* queue ne désigne aucune à remplir */
```

```
Debut
```

```
    f.tete := -1
```

```
    f.queue = -1
```

```
fin
```

Files

3. Implémentation sous forme de tableaux

- Test de la file vide
 - ❑ La file devient vide quand on prélève l'élément de la seule case du tableau encore remplie
 - ❑ Avant le retrait du dernier élément de la file, tete et queue désignent la même case
 - ❑ Après le retrait du dernier élément, on réinitialise tete et queue à -1

```
fonction file_vide(f:file):booleen
/* retourne vrai si la file est vide, faux sinon*/
debut
    retourner(f.tete = -1)
fin
```

Files

3. Implémentation sous forme de tableaux

- Test de la file pleine

```
fonction file_pleine(f :file):booleen  
/* retourne vrai si la file est pleine, faux sinon*/  
debut  
    retourner(f.tete=(f.queue+1)mod nbelt)  
fin
```


Files

3. Implémentation sous forme de tableaux

- Retrait de l'élément en tête de file

```
procedure retirer(var f :file, var v : entier, var ok : boleen)
/* retrait d'un élément en tête de file si celle-ci n'est pas vide* /
debut
    ok := non(file_vide(f))
    si ok alors
        v:= f.tab[f.tete]
        si f.tete <> f.queue alors
            f.tete:= (f.tete + 1)mod nbelt
        sinon
            init_file(f)
    fsi
fsi
fin
```

Files

3. Implémentation sous forme de tableaux

- Ajout d'un élément en fin de file

```
procedure ajouter(var f :file, v : entier, var ok : booleen)  
/* ajout de la valeur v en fin de file si celle-ci n'est pas pleine */  
debut  
    ok := non(pleine_file(f))  
    si ok alors  
        f.queue := (f.queue+1)mod nbelt  
        f.tab[f.queue]:= v  
        si f.tete=-1 alors  
            f.tete:=0  
        fsi  
    fsi  
fin
```