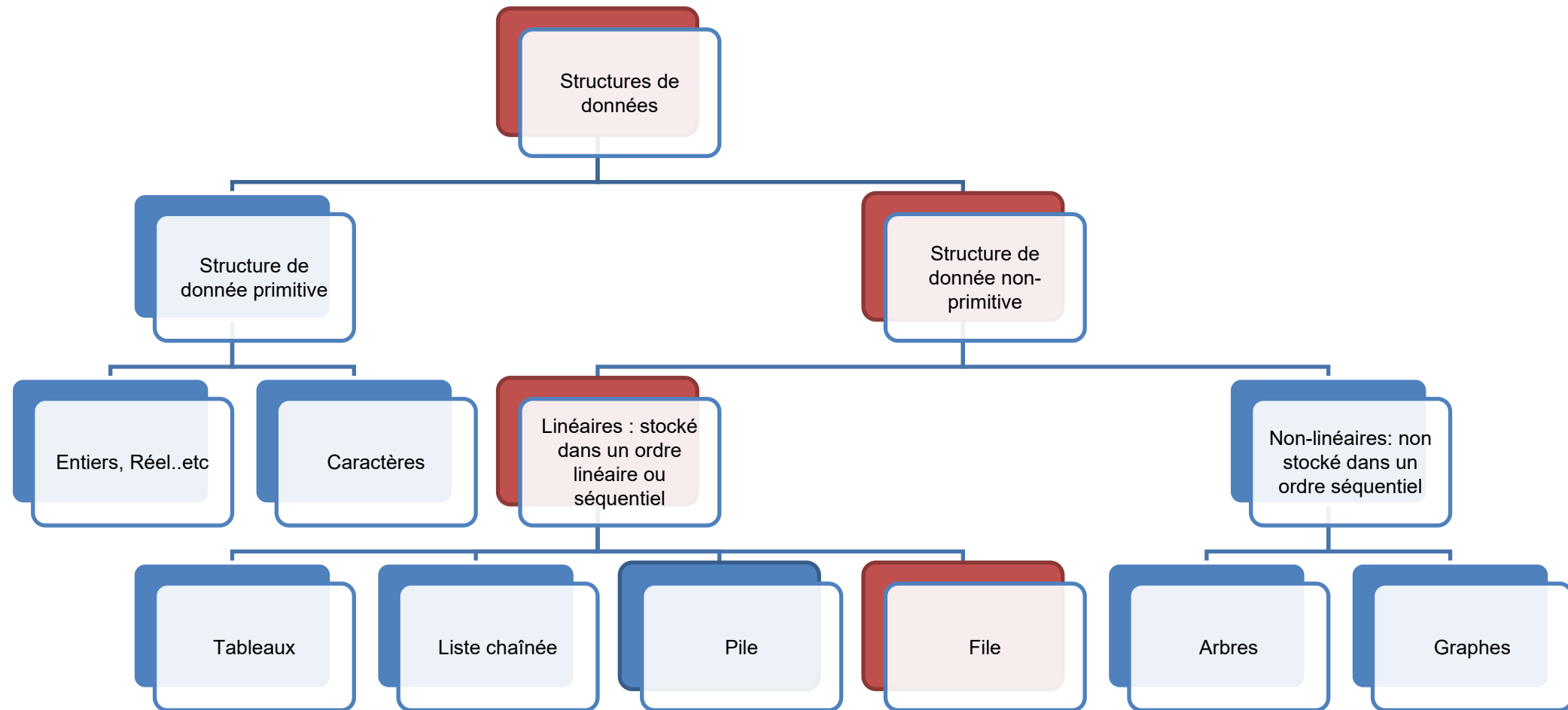


Programmation & Algorithmique II

CM 10 : Les Files (2)

CLASSIFICATION DES STRUCTURES DE DONNÉES

➤ Structures de données primitives et non primitives



PLAN

- Les Types de Files
 - File circulaire ✓
 - Exemple
 - Gestion circulaire par tableau
 - Primitives
 - File d'attente double (Dequeue)
 - Exemple
 - Primitives

PLAN

- Les Types de Files

- File circulaire

- Exemple

- Gestion circulaire par tableau

- Primitives

- File d'attente double (Dequeue)

- Exemple

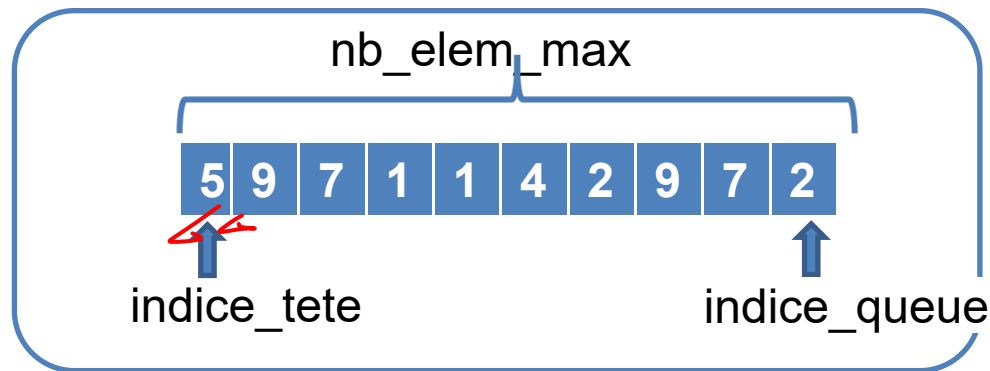
- Primitives

IMPLÉMENTATION SOUS FORME DE TABLEAU

> Gestion naïve

> Supposons l'exemple suivante :

***Enfiler(5); Enfiler(9); Enfiler(7); Enfiler(1); Enfiler(1);
Enfiler(4); Enfiler(2); Enfiler(9); Enfiler(7); Enfiler(2);***



Peut-on encore enfiler des éléments?

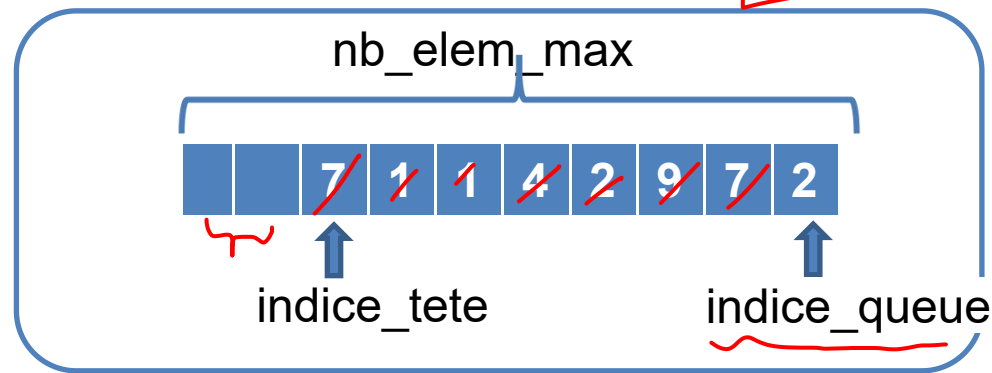
Non, car la file est pleine

IMPLÉMENTATION SOUS FORME DE TABLEAU

> Gestion naïve

> Ensuite :

Défiler(); Défiler();



Peut-on faire *Enfiler(3)*?

Non, impossible de faire *Enfiler(3)* car

(`F.indice_queue == F.nb_elem_max-1`) est **Vrai**

La file est toujours considérée comme pleine

REPRÉSENTATION D'UNE FILE

- **Solution : Représenter la file par tableau circulaire**
 - Les éléments de la file sont rangés dans un tableau
 - Deux indices sont nécessaires pour indiquer respectivement la tête et la queue de la file



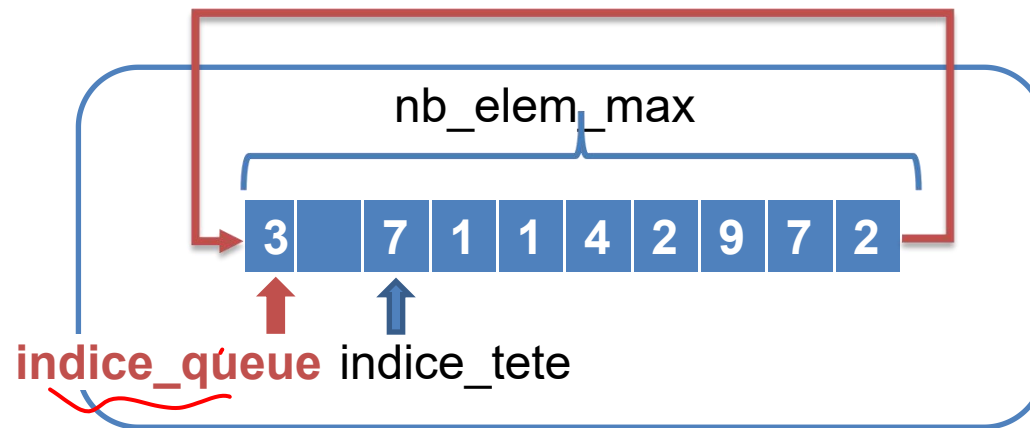
Ceci n'est pas totalement trivial : La difficulté provient du fait que, au cours des évolutions de la file, ces indices peuvent « faire le tour » du tableau, qui doit alors être considéré comme un anneau.



IMPLÉMENTATION SOUS FORME DE TABLEAU CIRULAIRE

> Gestion par tableau circulaire

Enfiler(3) est possible

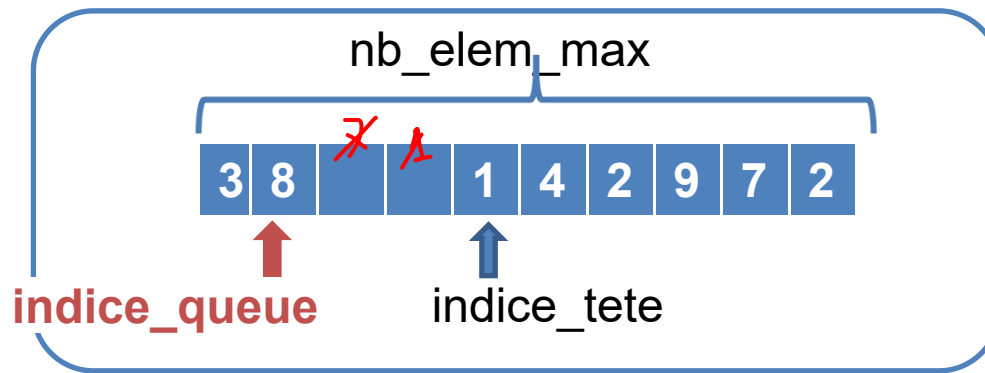


IMPLÉMENTATION SOUS FORME DE TABLEAU CIRULAIRE

> Gestion par tableau circulaire

> Ensuite :

Defiler(); Defiler (), Enfiler(8)



➔ Certaines primitives de gestion des Files doivent être modifiées

IMPLÉMENTATION SOUS FORME DE TABLEAU CIRULAIRE

Qu'est-ce qui va changer :

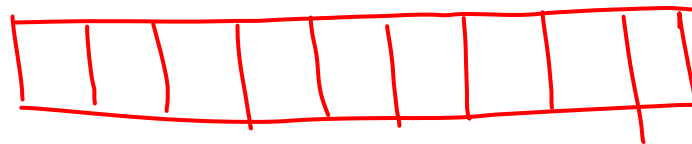
- Initialement, définissez la valeur de indice_tête à -1 (pour indiquer que la file est vide)
- Enfiler:
 - Si la file est vide alors mettre indice_tête à 0
 - Incrémenter indice_queue « circulairement »
- Défiler
 - Si la file contient un seul élément, alors mettre indice_tête et indice_queue à -1
 - Incrémenter indice_tête « circulairement »

IMPLÉMENTATION SOUS FORME DE TABLEAU CIRULAIRE

> Créer une file vide.

- > La fonction permettant de créer une file vide est la suivante :

```
File Initialiser(int nb_max)
{
    File filevide;
    filevide.indice_tete=-1; /* la file est vide */
    filevide.indice_queue=-1;
    filevide.nb_elem_max = nb_max; /* capacité nb_max */
    /* allocation des éléments : */
    filevide.tab = (TypeDonnee*)malloc(nb_max*sizeof(TypeDonnee));
    return filevide;
}
```



Type Donnee [nb_max]

IMPLÉMENTATION SOUS FORME DE TABLEAU CIRULAIRE

> File vide,

- > La fonction permettant de savoir si la file est vide est la suivante. La fonction renvoie
 - > 1 si le nombre d'éléments est égal à 0.
 - > La fonction renvoie 0 dans le cas contraire.

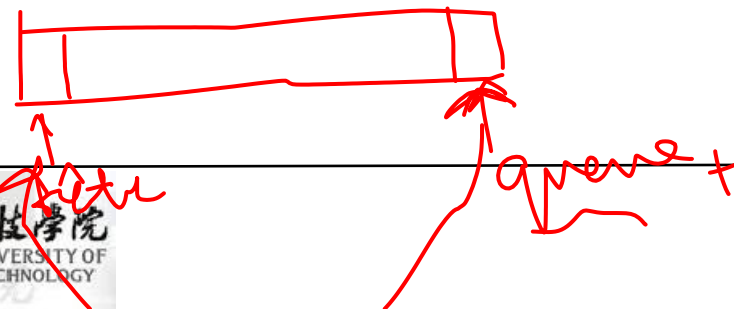
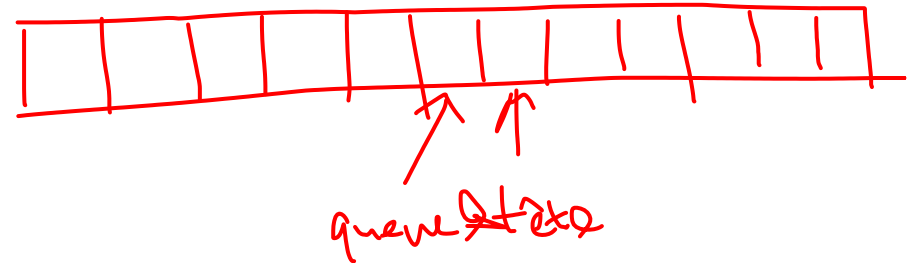
```
int EstVide(File F)
{
    if (F.indice_tete == -1)
        return 1;
    else
        return 0;
}
```

IMPLÉMENTATION SOUS FORME DE TABLEAU CIRULAIRE

> File pleine

- > La fonction permettant de savoir si la file est pleine est la suivante :

```
int EstPleine(File F)
{
    if (F.indice_tete == (F.indice_queue+1)%F.nb_elem_max)
        return 1;
    else
        return 0;
}
```



IMPLÉMENTATION SOUS FORME DE TABLEAU CIRULAIRE

> Accéder à la tête de la file

- > La fonction effectue un passage par adresse pour ressortir le tête de la file
 - > La tête de la file est le premier élément entré, qui est l'élément du tableau avec l'indice indice_tete
- > La fonction permet d'accéder à la tête de la file et renvoie le code d'erreur 1 en cas de liste vide et 0 sinon

```
int AccederTete(File F, TypeDonnee *pelem)
{
    if (EstVide(F))
        return 1; /* on retourne un code d'erreur */
    *pelem = F.tab[F.indice_tete]; /* on renvoie l'élément */
    return 0;
}
```

IMPLÉMENTATION SOUS FORME DE TABLEAU CIRULAIRE

➤ Ajouter un élément (*Enfiler*)

- Pour modifier le nombre d'éléments de la file, il faut passer la file par adresse. La fonction Enfiler, qui renvoie 1 en cas d'erreur et 0 dans le cas contraire, est la suivante :

```
int Enfiler(File *pF, TypeDonnee elem)
{
    if (EstPleine(*pF))
        return 1; /* on ne peut rien ajouter à une file pleine */
    /*
    pF->indice_queue++; /* insertion en queue de file */
    if (pF->indice_tete == -1) /* si file vide */
        pF->indice_tete = 0;
    if (pF->indice_queue == pF->nb_elem_max) /* si au bout */
        pF->indice_queue = 0; /* on réutilise le début */
    pF->tab[pF->indice_queue] = elem; /* ajout de l'élément */
    /*
    return 0;
    */
}
```

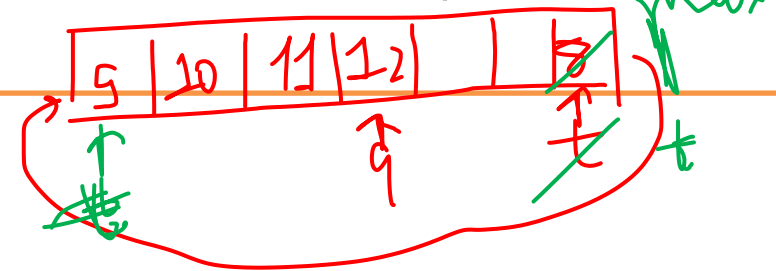
Diagram illustrating a circular queue with 6 slots. The first slot contains the value 3. Red arrows indicate the movement of the queue pointers: $q = \text{max} - 1$ and $q + 1 = \text{max}$. A red arrow labeled "queue" points to the first slot. A red arrow labeled "tete" points to the last slot.

IMPLÉMENTATION SOUS FORME DE TABLEAU CIRULAIRE

➤ Supprimer un élément (*Defiler*)


- La fonction Defiler supprime la tête de la file en cas de file non vide. La fonction renvoie 1 en cas d'erreur (file vide), et 0 en cas de succès.

```
int Defiler(File *pF, TypeDonnee *pelem)
{
    if (EstVide(*pF))
        return 1; /* erreur : file vide */
    *pelem = pF->tab[pF->indice_tete]; /* renvoie l'élément */
    if (pF->indice_tete == pF->indice_queue) /* si la file
        contient un seul élément */
        pF->indice_tete = pF->indice_queue = -1;
    else
        pF->indice_tete++;
    if (pF->indice_tete == pF->nb_elem_max) /* si on est au
        bout */
        pF->indice_tete = 0; /* on passe au début du tableau */
    return 0;
}
```



IMPLÉMENTATION SOUS FORME DE TABLEAU CIRULAIRE

> Vider et détruire



```
void Vider(File *pF)
{
    pF->indice_tete = -1; /* réinitialisation des indices */
    pF->indice_queue = -1;
}
void Detruire(File *pF)
{
    if (pF->nb_elem_max != 0)
        free(pF->tab); /* libération de mémoire */
    pF->nb_elem_max = 0; /* file de taille 0 */
}
```

QUIZ



Dans une file d'attente, l'opération de suppression d'un élément (Defiler) est possible si la file :

- A- N'est pas pleine
- B- N'est pas vide
- C- N'est ni vide, ni pleine

5 minute

PLAN

- Les Types de Files
 - File circulaire
 - Exemple
 - Gestion circulaire par tableau
 - Primitives
 - File d'attente double (Deque)
 - Exemple
 - Primitives

FILE D'ATTENTE DOUBLE (DEQUE)

- Une file d'attente double (en anglais Double-ended queue), appelée « deque » est un type de données abstrait qui généralise une file d'attente
- Dans une Deque les éléments peuvent être ajoutés ou supprimés de l'avant (tête) ou de l'arrière (queue)



La Deque ne suit pas la règle FIFO

File + Pile

FILE D'ATTENTE DOUBLE (DEQUE)

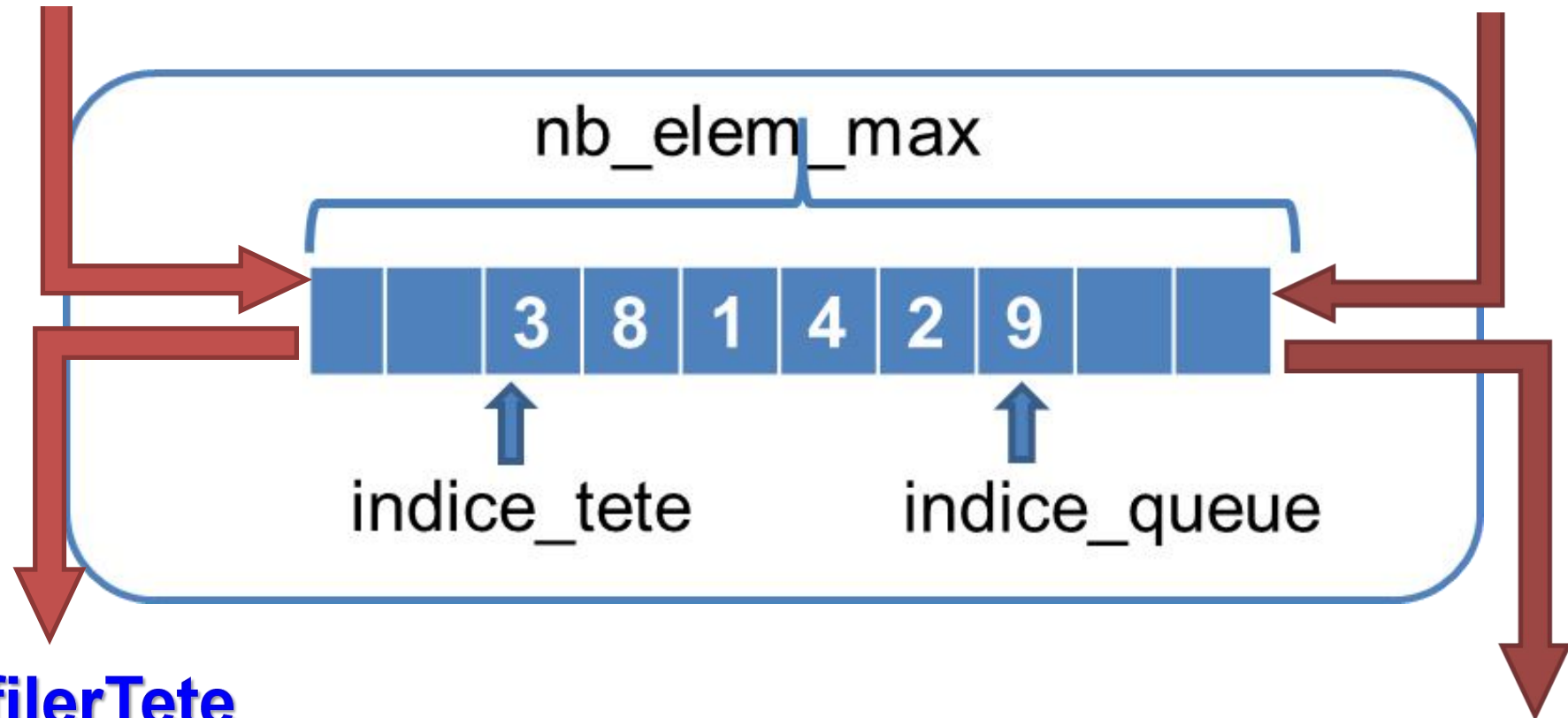
Les primitives de File d'attente double (Deque)

- > **Initialiser** : cette fonction crée une file vide.
- > **EnfilerTete** : cette fonction permet d'ajouter un élément à la tête de la file.
- > **EnfilerQueue** : cette fonction permet d'ajouter un élément à la queue de la file.
- > **DefilerTete** : cette fonction supprime le début de la file. L'élément supprimé est retourné par la fonction Defiler pour pouvoir être utilisé.
- > **DefilerQueue** : cette fonction supprime l'élément se trouvant à la fin de la file. L'élément supprimé est retourné par la fonction Defiler pour pouvoir être utilisé.
- > ...

FILE D'ATTENTE DOUBLE (DEQUEUE)

EnfilerTete

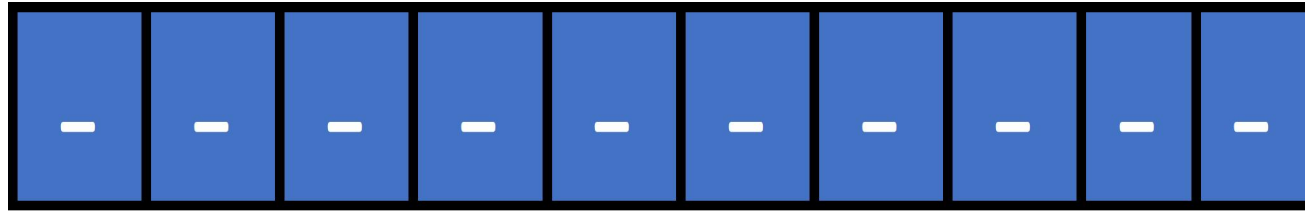
EnfilerQueue



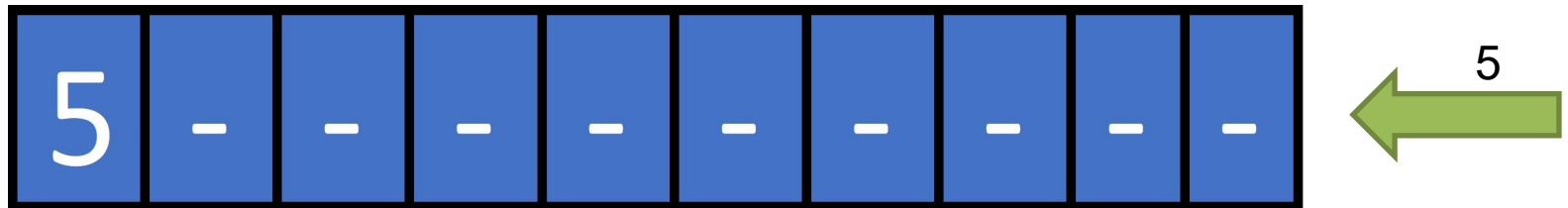
DefilerQueue

FILE D'ATTENTE DOUBLE (DEQUE)

DEQUE Vide

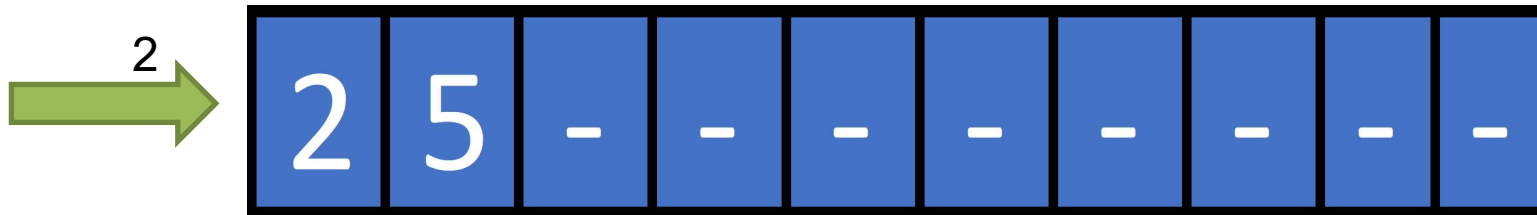


EnfilerQueue(5):

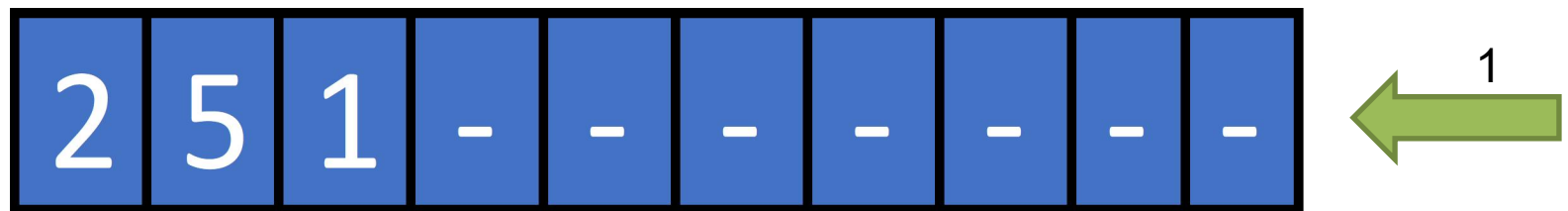


FILE D'ATTENTE DOUBLE (DEQUEUE)

EnfilerTete(2):

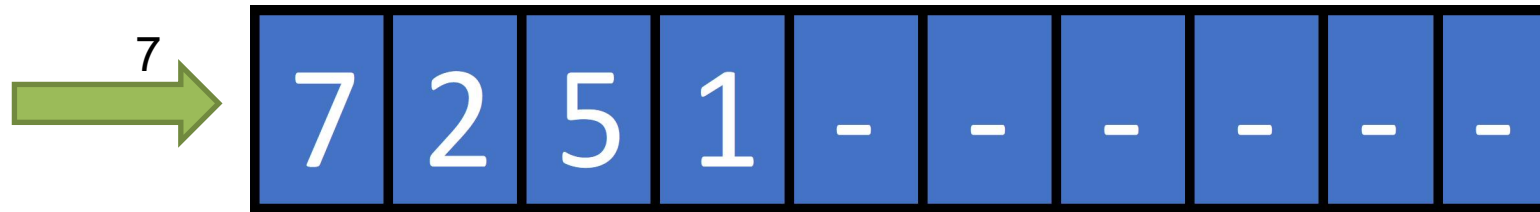


EnfilerQueue(1):

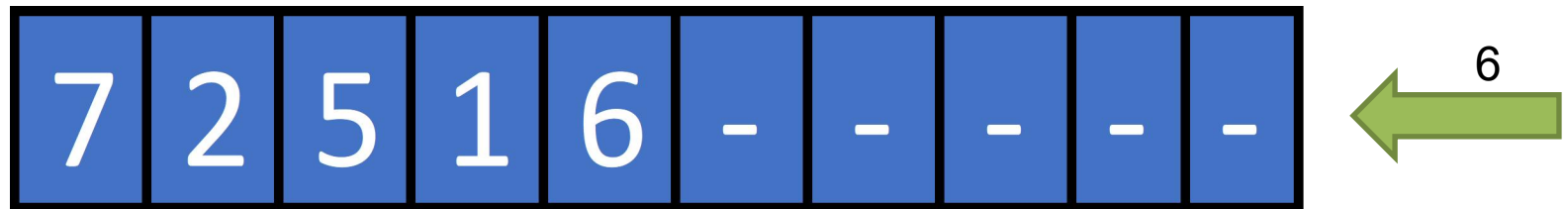


FILE D'ATTENTE DOUBLE (DEQUEUE)

EnfilerTete(7):

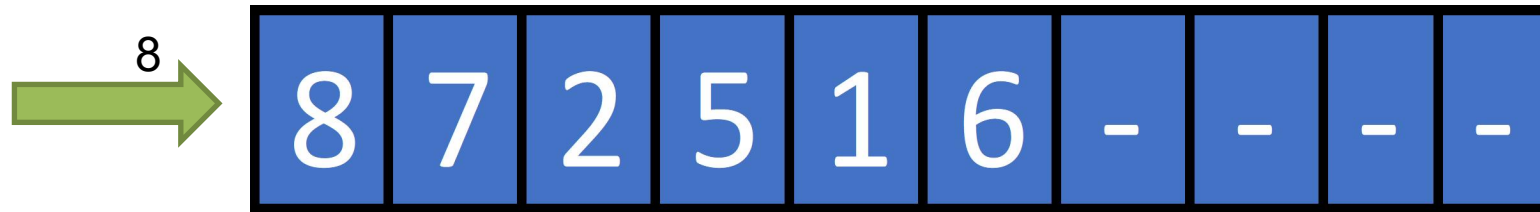


EnfilerQueue(6):

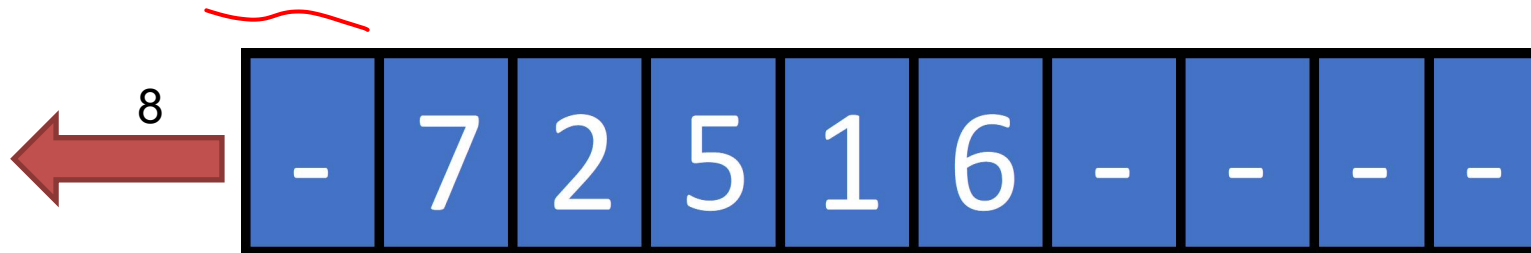


FILE D'ATTENTE DOUBLE (DEQUE)

EnfilerTete(8):



DefilerTete() --> 8

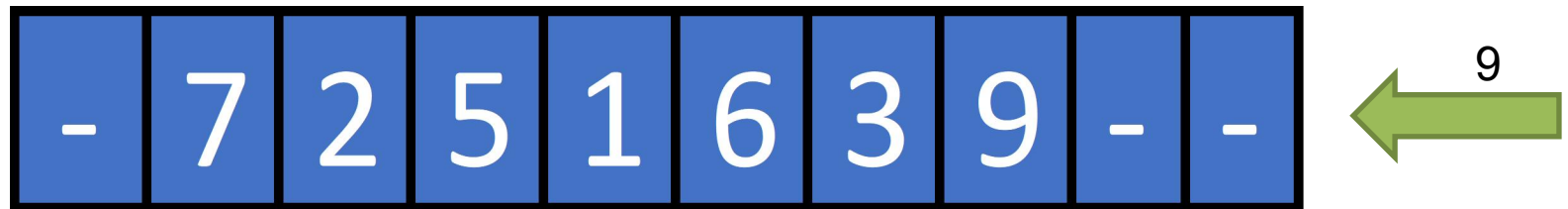


FILE D'ATTENTE DOUBLE (DEQUEUE)

EnfilerQueue(3):

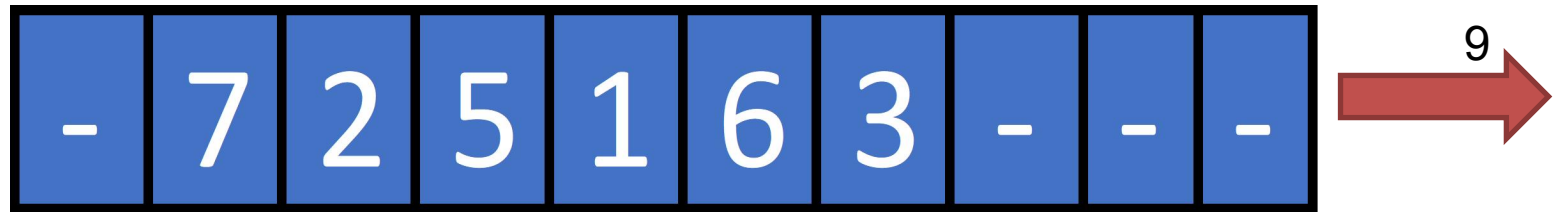


EnfilerQueue(9):



FILE D'ATTENTE DOUBLE (DEQUEUE)

DefilerQueue() --> 9



FILE D'ATTENTE DOUBLE (DEQUE)

- Pour implémenter une File d'attente double (Deque) sous forme de tableau, on crée la structure de données suivante.


```
typedef int TypeDonnee;  
typedef struct  
{  
    int nb_elem_max; /* nombre d'éléments maximum*/  
    int nb_elem; /* nombre d'éléments*/  
    int indice_tete, indice_queue;  
    TypeDonnee *tab; /* tableau des éléments */  
} File;
```

FILE D'ATTENTE DOUBLE (DEQUE)

> Créer une Deque vide.

- > La fonction permettant de créer une file vide est la suivante :

```
File Initialiser(int nb_max)
{
    File filevide;
    filevide.indice_tete=-1; /* la pile est vide */
    filevide.indice_queue=-1;
    filevide.nb_elem_max = nb_max; /* capacité nb_max */
    filevide.nb_elem = 0;
    /* allocation des éléments : */
    filevide.tab =
(TypeDonnee*)malloc(nb_max*sizeof(TypeDonnee));
    return filevide;
}
```



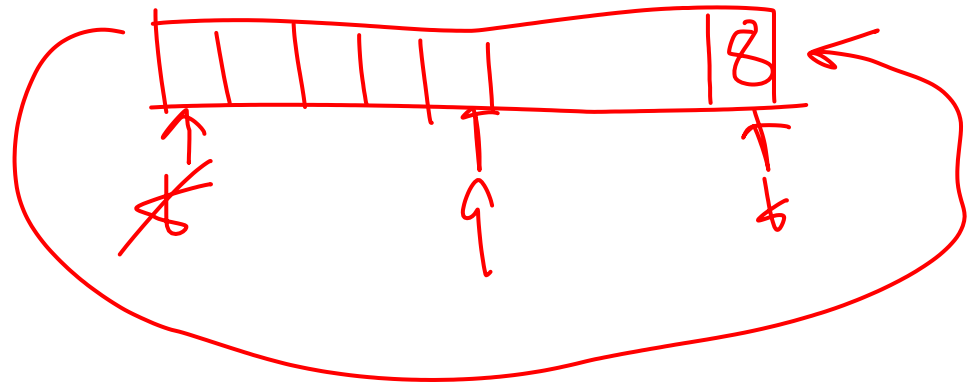
DEQUE: AJOUTER UN ÉLÉMENT À LA TÊTE (AU DÉBUT)

```
void EnfilerTete(File *pF, int element) {
    int i, k, c;

    if (pF->indice_tete == 0 && pF->indice_queue == pF->nb_elem_max - 1) {
        printf("\nDeque est pleine.\n");
        return;
    }
    pF->nb_elem++;
    if (pF->indice_tete == -1) {
        pF->indice_tete = pF->indice_queue = 0;
        pF->tab[pF->indice_tete] = element;
        return;
    }

    /* Implantation circulaire*/
    if (pF->indice_tete == 0) { /* la pile n'est pas plein donc il y a de la place à la fin */
        pF-> indice_tete = pF->nb_elem_max - 1;
        pF->tab[pF->indice_tete] = element;
    } else {
        .....
    }

    else {
        (pF->indice_tete)--;
        pF->tab[pF->indice_tete] = element;
    }
}
```



DEQUE: AJOUTER UN ÉLÉMENT À LA QUEUE (À LA FIN)

```
void EnfilerQueue(File *pF, int element) {
    int i, k;

    if (pF->indice_tete == (pF->indice_queue + 1) % pF->nb_elem_max) {
        printf("\nDeque est pleine.\n");
        return;
    }
    pF->nb_elem++;

    if (pF->indice_tete == -1) {
        pF->indice_queue = pF->indice_tete = 0;
        pF->tab[pF->indice_queue] = element;
        return;
    }

    /* Implantation circulaire*/
    if (pF-> indice_queue == pF->nb_elem_max - 1) { /* la pile n'est pas plein donc il y a de
    la place à la tête */
        pF-> indice_queue = 0;
    } else {
        (pF->indice_queue)++;
    }
    pF->tab[pF->indice_queue] = element;
}
```


DEQUE: SUPPRIMER UN ÉLÉMENT À LA TÊTE (AU DÉBUT)

```
int DefilerTete(File *pF) {  
    int element;
```

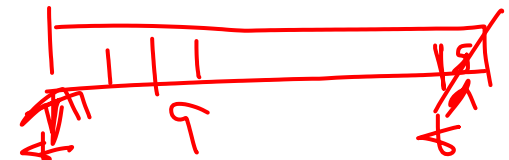
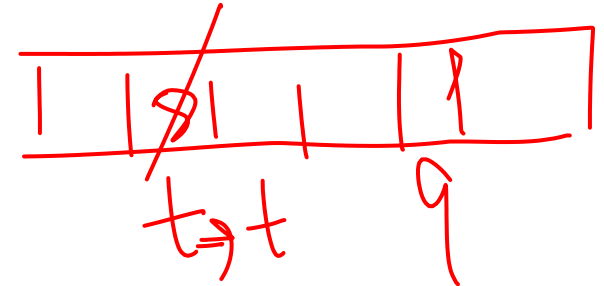
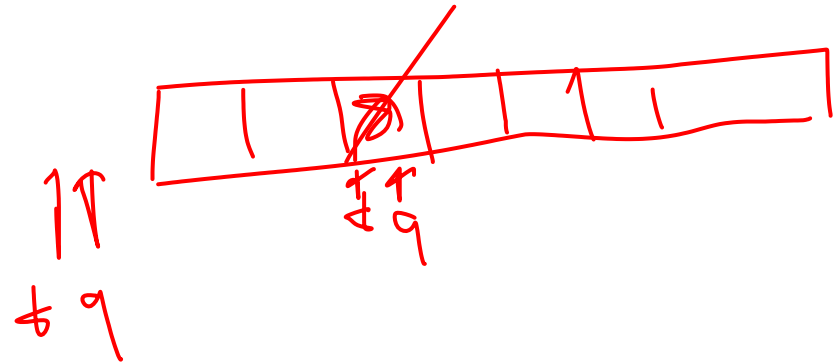
```
    if (pF->indice_tete == -1) {  
        printf("\nDeque est vide.\n");  
        return 0;  
    }
```

```
    pF->nb_elem--;
```

```
    element = pF->tab[pF->indice_tete];  
    pF->tab[pF->indice_tete] = 0;
```

```
    if (pF->indice_tete == pF->indice_queue) 1 element  
        pF->indice_tete = pF->indice_queue = -1;
```

```
    else  
        /* Implantation circulaire */  
        if (pF->indice_tete == pF->nb_elem_max - 1) { /* l'indice de la tête  
est à la fin du tableau */  
            pF->indice_tete = 0;  
        } else {  
            (pF->indice_tete)++;  
        }  
    }
```



DEQUE: SUPPRIMER UN ÉLÉMENT À LA QUEUE (À LA FIN)

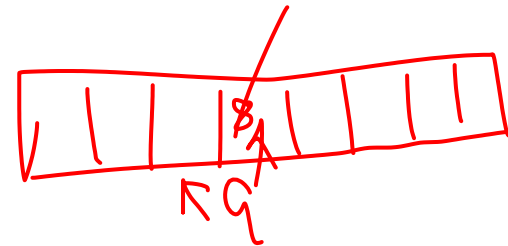
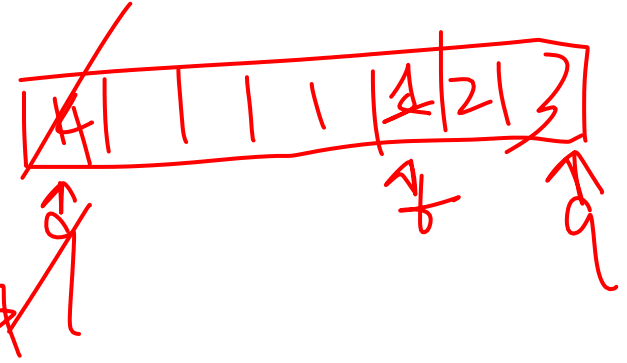
```
int DefilerQueue(File *pF) {
    int element;

    if (pF->indice_tete == -1) {
        printf("\nDeque est vide.\n");
        return 0;
    }

    pF->nb_elem--;

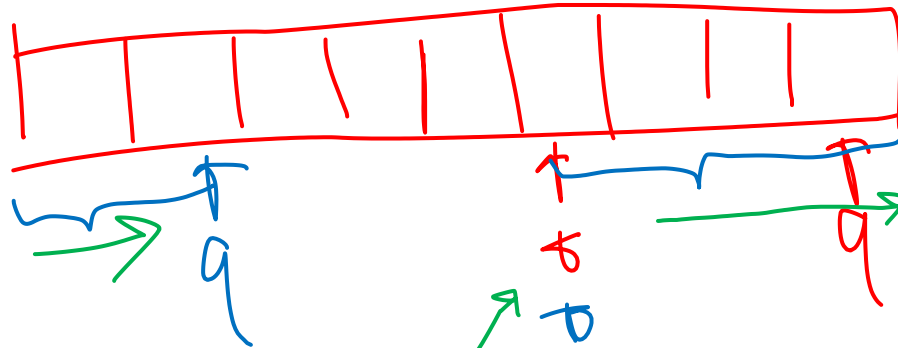
    element = pF->tab[pF->indice_queue];
    pF->tab[pF->indice_queue] = 0;

    /* Implantation circulaire */
    if (pF->indice_queue == 0) { /* l'indice de la queue est au début du
    tableau */
        pF->indice_queue = pF->nb_elem_max - 1;
    } else {
        (pF->indice_queue)--;
    }
    return element;
}
```



EXERCICES

- Écrire la fonction qui affiche les éléments d'une file d'attente représentée avec un tableau circulaire



$\text{si } t < q : t \rightarrow q$
 $\text{si } q < t : t \rightarrow \text{fin} \rightarrow \text{débüt} \rightarrow \text{queue}$

if estVide(f): return;
 else if

if $t < q$
 for

(int i = t; i <= q; i++)

print f->tab[i]

else

for (int i = t,

$i < \text{nb_de_max}$

$i++$

print tab[i]

for (int i = 0; i <= q; i++)
 print tab[i]

EXERCICES

- Écrire la fonction qui affiche les éléments d'une file d'attente représentée avec un tableau circulaire

```
void Afficher(File F){
    if ( EstVide (F))
        return;
    if (F.indice_queue >= F.indice_tete){
        for (int i = F.indice_tete; i <= F.indice_queue; i++)
            printf(" %.0f <-", F.tab[i]);
    }
    else{
        for (int i = F.indice_tete; i < F.nb_elem_max; i++)
            printf(" %.0f <-", F.tab[i]);

        for (int i = 0; i <= F.indice_queue; i++)
            printf(" %.0f <-", F.tab[i]);
    }
    printf("\n");
}
```