Algorithmique & Structures de données

Structure de file d'attente

hepia, HES-SO//Genève ITI

1. Définition

Une file d'attente ou queue est une structure de données dans laquelle la première information entrée est la première à ressortir (en anglais : *FIFO, First In First Out*)
On considère ici une implémentation de queue dynamique et donc à capacité « infinie » (dépend en réalité de la mémoire disponible).

2. Spécification de la structure

Si le type des éléments qu'on insère, n'est pas précisé, on parle de spécification générique.
Les opérations possibles sur une file d'attente (ou queue) en définissent l'interface :

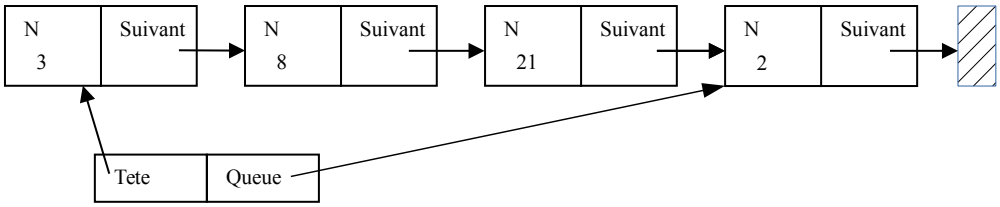
- Opérations de manipulation
  - Insérer un nouvel élément en queue de la file d'attente
  - Extraire un élément en tête de la file d'attente (i.e. le premier élément introduit)
- Fonctions de consultation
  - Lire l'élément en tête de la file d'attente
  - Lire l'élément en queue de la file d'attente
  - Tester si la file d'attente est vide

L'interface de la file d'attente est l'ensemble des fonctionnalités offertes à l'utilisateur pour interagir avec celle-ci.

3. Structure de file d'attente

On considère une liste chaînée d'articles avec deux pointeurs de tête et de queue. Chaque article comportera un champ pour stocker les valeurs dans cette liste et un champ contenant une variable de type accès pour assurer le chaînage. On considère à titre d'exemple le cas de valeurs entières.

Schématiquement:



L'accès à la file d'attente se fera par les pointeurs de tête et de queue.

Une telle file d'attente basée sur une liste chaînée sera déclarée par exemple sous la forme suivante:

```
type T_Element;                      -- Pré-déclaration indispensable !
type T_Lien is access T_Element;     -- Pointeur sur des éléments de liste
type T_Element is record             -- Elément de liste
   N : Integer ;
   Suivant : T_Lien := null;
end record;
type T_Queue is record               -- File d'attente
   Tete  : T_Lien := null;
   Queue : T_Lien := null;
end record;
QUEUE_VIDE : exception;
```

4. Implémentation des fonctionnalité d'une file d'attente

a) Consultations

On considère la fonction `Vide` qui teste si la file d'attente est vide et les fonctions `Tete` et `Queue` qui retournent l'élément en tête, respectivement en queue, de la file d'attente.

```
function Vide(FA : T_Queue) return Boolean is
begin
   return FA.Tete = null and FA.Queue = null;
end Vide;
function Tete(FA : T_Queue) return Integer is
begin
   if Vide(FA) then
      raise QUEUE_VIDE;
   end if;
   return FA.Tete.N;
end Tete;
function Queue(FA : T_Queue) return Integer is
begin
   if Vide(FA) then
      raise QUEUE_VIDE;
   end if;
   return FA.Queue.N;
end Queue;
```

b) Manipulations

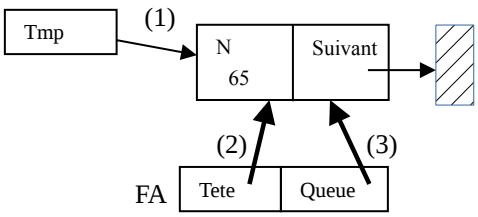
Insertion en queue de file d'attente

Voici l'entête de la procédure :

```
procedure Inserer(FA : in out T_Queue; Val : in Integer);
```

Il faut considérer plusieurs cas :

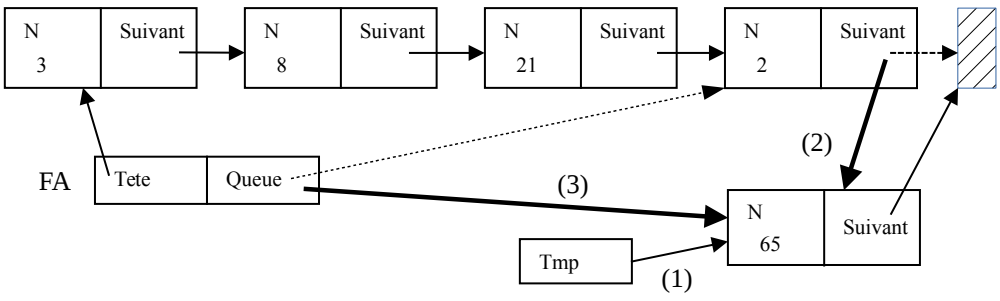
a) La file d'attente est vide




```
(1) Tmp := new T_Element'(65,null);
(2) FA.Tete := Tmp;
(3) FA.Queue := Tmp;
```

b) L'insertion se fait en queue d'une file d'attente non vide

```
(1) Tmp := new T_Element'(65,null);
(2) FA.Queue.Suivant := FA.Queue;
(3) FA.Queue := Tmp
```



Extraction en tête de file d'attente

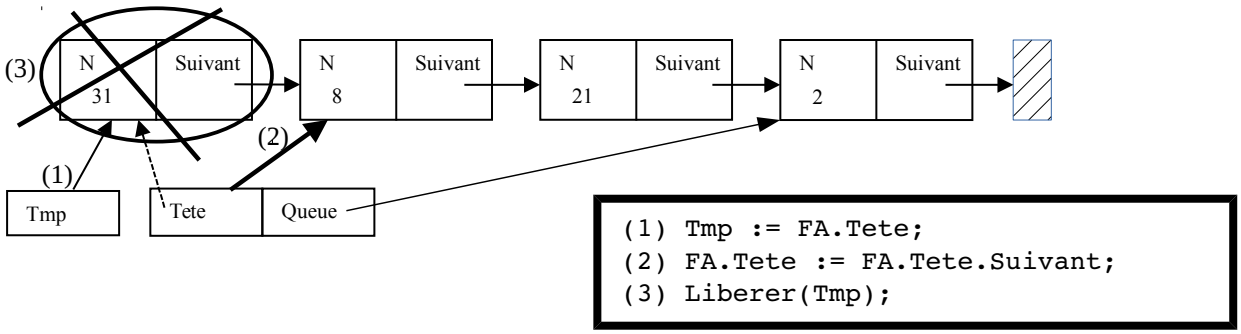
Voici l'entête de la procédure :

```
procedure Extraire(FA : in out T_Queue; Val : out Integer);
```

Si la liste est vide, on lève l'exception `QUEUE_VIDE` via un appel : `raise QUEUE_VIDE;`

Sinon on commence par récupérer, la valeur en tête de file d'attente via : `Val := FA.Tete.N;`

puis on met un pointeur temporaire sur l'élément en tête, avant de déplacer le pointeur de tête sur l'élément suivant. Finalement, on désalloue la mémoire.



La procédure `Liberer` est obtenue par instanciation à partir du paquetage `Ada.Unchecked_Deallocation` via la déclaration :

```
procedure Liberer is new Ada.Unchecked_Deallocation(T_Element,T_Lien);
```

Si la file d'attente ne contenait qu'un seul élément, alors il faut mettre le pointeur `FA.Queue` à `null`.

Dans ce cas, à la suite du point (3), le pointeur `FA.Tete` se retrouve à `null`. On doit donc ajouter l'instruction :

```
if FA.Tete = null then
   FA.Queue := null;
end if;
```