

# Analyse et programmation langage ADA

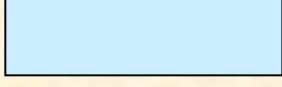
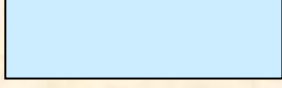


Informatique 1<sup>ère</sup> année

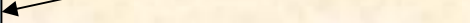
# Pointeurs et structures de données avancées

- Liste
- File statique
- File dynamique
- Pile statique
- Pile dynamique

# Liste



**Tête**



**Element précédent**



**Elément courant**



**Element suivant**



**Queue**

- **STRUCTURES LINEAIRES SIMPLES TYPE ACCES**
  - **structures de données** (*data structures*)
  - algorithmes de **recherche** (*searching*);
  - **pointeurs** (*pointers*).
  
- **Listes, files**
  - **tête** (*head*)
  - **queue** (*tail*)
  - **élément suivant** (*next element*)
  - **élément précédent** (*previous element*)
  - **élément courant** (*current element*).

- Manipuler des listes:
  - **insérer** (*insert*)
  - **supprimer** (*delete*) ou **extraire** (*extract*)
  - **modifier** (*modify*);
  - **consulter** (*read*);
  - savoir si la liste est **vide** (*empty*);
  - **rechercher** (*search*)
  - **parcourir** (*traverse*)

- **Gestion, manipulations de base**

- si l'insertion se passe en queue et la suppression en tête (ou inversement), alors la liste s'appelle une **queue** (*queue*) (*first in, first out* abrégé *FIFO*)
- si l'insertion et la suppression ont lieu toutes les deux en tête (ou en queue), alors la liste s'appelle une **pile** (*stack*); (*last in, first out* abrégé *LIFO*)
- si l'insertion s'effectue selon un critère d'ordre et la suppression en tête, alors la liste s'appelle une **queue de priorité** (*priority queue*).

- **Liste statique**
- Une liste statique est donc une structure de longueur maximale connue à la compilation

```

Longueur_Max : constant := 100;                -- Longueur maximum d'une liste
subtype T_Longueur is Integer range 0..Longueur_Max;
subtype T_Numerotation is Integer range 1..Longueur_Max;
type T_Contenu is array ( T_Numerotation ) of T_Info;
type T_Liste_Statique is                    -- Pour une liste statique
record
  -- Contenu de la liste
  Contenu : T_Contenu;
  Longueur : T_Longueur := T_Longueur'First;  -- Longueur de la liste
  Tete : T_Numerotation;                      -- Tete et Queue de la liste
  Queue : T_Numerotation;
end record;

```

# Liste dynamique simplement chaînée

Exemple 1 :

**type** Element;

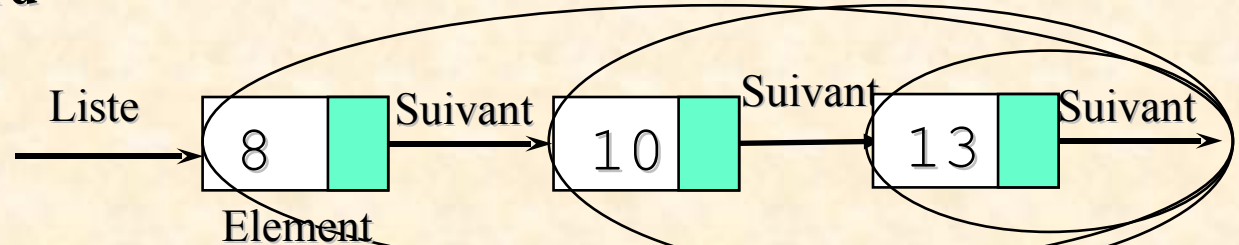
**type** Liste **is access** Element;

**type** Element **is record**

Info : Integer;

Suivant : Liste;

**end record;**



Exemple 2 :

**type** Element;

**type** Lien **is access** Element;

**type** Element **is record**

Info : Integer;

Suivant : Lien;

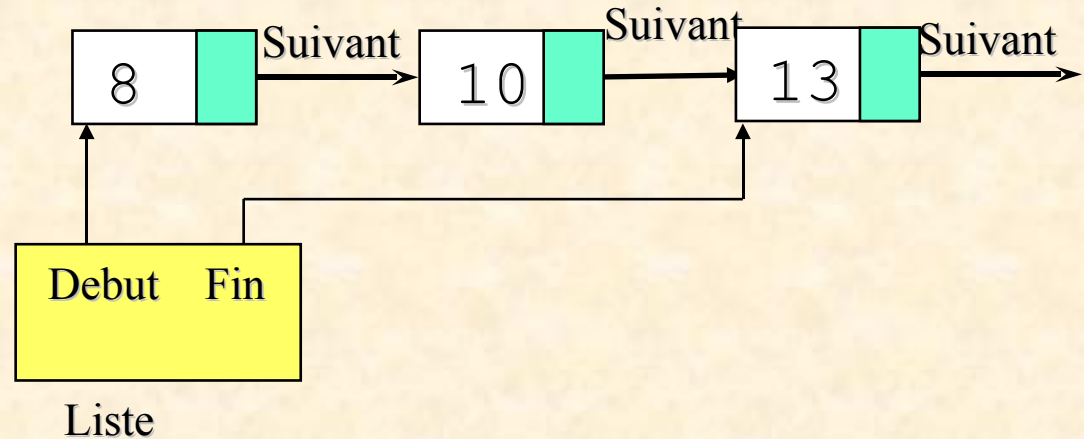
**end record;**

**type** Liste **is record**

Debut : Lien;

Fin : Lien;

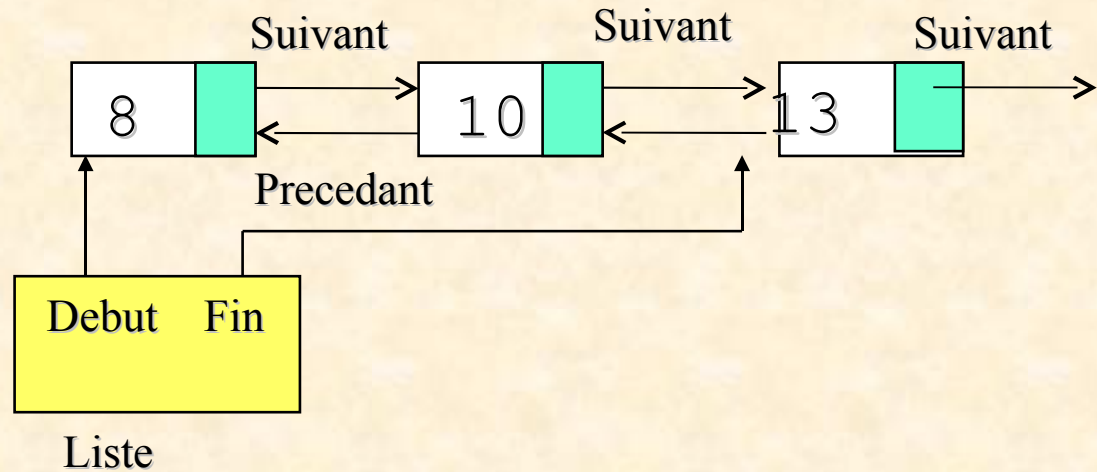
**end record;**





# Liste dynamique doublement chaînée

```
type Element;  
type Lien is access Element;  
type Element is record  
  Info : Integer;  
  Suivant : Lien;  
  Precedant : Lien;  
end record;  
type Liste is record  
  Debut : Lien;  
  Fin : Lien;  
end record;
```

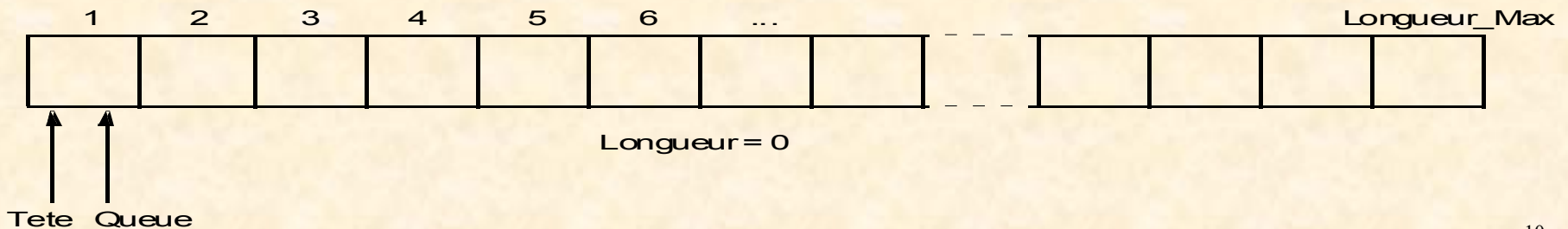


- **Queues statiques**

```

Longueur_Max : constant := 100; -- Longueur maximum d'une queue
subtype T_Longueur is Integer range 0..Longueur_Max;
subtype T_Numerotation is Integer range 1..Longueur_Max;
subtype T_Pos_Indic is Integer range 1..Longueur_Max + 1;
type T_Contenu is array ( T_Numerotation ) of T_Info;           --*
type T_Queue_Statique is -- Pour une queue statique
  record
    -- Contenu de la queue
    Contenu : T_Contenu;                                           --*
    Longueur : T_Longueur := T_Longueur'First; -- Longueur de la queue
    Tete : T_Pos_Indic := T_Pos_Indic'First;   -- Tete de la queue
    Queue : T_Pos_Indic := T_Pos_Indic'First; -- Queue de la queue
  end record;

```



```

package Queues_Statiques is
  type T_Info is ...;  -- Depend de l'application
  Longueur_Max : constant := 100;  -- Longueur maximum d'une queue
  subtype T_Longueur is Integer range 0..Longueur_Max;
  subtype T_Numerotation is Integer range 1..Longueur_Max;
  subtype T_Pos_Indic is Integer range 1..Longueur_Max + 1;
  type T_Contenu is array ( T_Numerotation ) of T_Info;  --*
  type T_Queue_Statique is  -- Pour une queue statique
  record
    -- Contenu de la queue
    Contenu : T_Contenu;
  --*
    Longueur : T_Longueur := T_Longueur'First;  -- Longueur de la queue
    Tete : T_Pos_Indic := T_Pos_Indic'First;  -- Tete de la queue
    Queue : T_Pos_Indic := T_Pos_Indic'First;  -- Queue de la queue
  end record;
  Queue_Vide : exception;  -- Levee si la queue est vide
  Queue_Pleine : exception;  -- Levee si la queue est pleine
  -----
  -- Insere Info en queue de La_Queue. Leve Queue_Pleine si l'insertion
  -- est impossible (la queue contient deja Longueur_Max elements)
  procedure Insérer (La_Queue : in out T_Queue_Statique; Info : in T_Info );

```

```

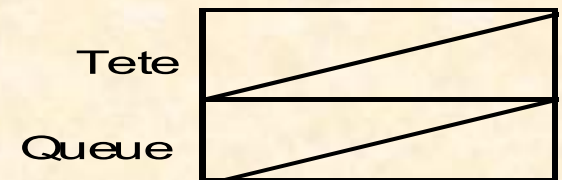
-- Supprime l'information (en tete de La_Queue) qui est rendue dans
-- Info. Leve Queue_Vide si la suppression est impossible (la queue est vide)
procedure Supprimer (La_Queue : in out T_Queue_Statique; Info : out T_Info );
-- Change l'information en tete de Queue. Leve Queue_Vide si la
-- modification est impossible (la queue est vide)
procedure Modifier (La_Queue : in out T_Queue_Statique; Info : in T_Info );
-- Retourne l'information en tete de La_Queue. Leve Queue_Vide si la
-- consultation est impossible (la queue est vide)
function Tete ( La_Queue : T_Queue_Statique) return T_Info;
-----
-- Retourne True si La_Queue est vide, False sinon
function Vide ( La_Queue : T_Queue_Statique) return Boolean;
-----
-- Retourne True si l'information Info est presente dans la queue,
-- False sinon
function Recherche (La_Queue : T_Queue_Statique; Info : T_Info) return Boolean;
-----
-- Effectue un traitement sur tous les elements de la queue
procedure Parcourir ( La_Queue : in out T_Queue_Statique );
end Queues_Statiques;

```

- **Queues dynamiques**

```
type T_Element;           -- Predeclaration
type T_Lien is access T_Element; -- Le type acces
type T_Element is       -- Pour une variable dynamique
  record
    Information : T_Info;  -- L'information, ici sur un seul champ
    Suivant : T_Lien;     -- Pour reperer l'element qui suit
  end record;
type T_Queue_Dynamique is -- Pour une queue dynamique
  record
    Tete : T_Lien;        -- Tete et queue de la queue
    Queue : T_Lien;
  end record;
```

Queue dynamique



-- Bases de gestion de queues dynamiques

**package** Queues\_Dynamiques **is**

**type** T\_Info **is** ...; -- Depend de l'application

**type** T\_Element; -- Predeclaration

**type** T\_Lien **is access** T\_Element; -- Le type acces

**type** T\_Element **is** -- Pour une variable dynamique

**record**

Information : T\_Info; -- L'information sur un seul champ

Suivant : T\_Lien; -- Pour reperer l'element qui suit

**end record;**

**type** T\_Queue\_Dynamique **is** -- Pour une queue dynamique

**record**

Tete : T\_Lien; -- Tete et queue de la queue

Queue : T\_Lien;

**end record;**

Queue\_Vide : **exception;** -- Levee si la queue est vide

-----  
-- Insere Info en queue de La\_Queue.

**procedure** Insérer (La\_Queue : **in out** T\_Queue\_Dynamique; Info : **in** T\_Info );

-----  
-- Supprime l'information (en tete de La\_Queue) qui est rendue dans

-- Info. Leve Queue\_Vide si la suppression est impossible (queue vide)

**procedure** Supprimer (La\_Queue : **in out** T\_Queue\_Dynamique; Info : **out** T\_Info );



```
-----  
-- Change l'information en tete de La_Queue. Leve Queue_Vide si la  
-- modification est impossible (la queue est vide)  
procedure Modifier (La_Queue : in T_Queue_Dynamique; Info : in T_Info );
```

```
-----  
-- Retourne l'information en tete de La_Queue. Leve Queue_Vide si la  
-- consultation est impossible (la queue est vide)  
function Tete ( La_Queue : T_Queue_Dynamique) return T_Info;
```

```
-----  
-- Retourne True si La_Queue est vide, False sinon  
function Vide ( La_Queue : T_Queue_Dynamique) return Boolean;
```

```
-----  
-- Retourne True si l'information Info est presente dans la queue,  
-- False sinon  
function Recherche (La_Queue : T_Queue_Dynamique; Info : T_Info) return Boolean;
```

```
-----  
-- Effectue un traitement sur tous les elements de la queue  
procedure Parcourir ( La_Queue : in T_Queue_Dynamique );
```

```
-----  
end Queues_Dynamiques;
```

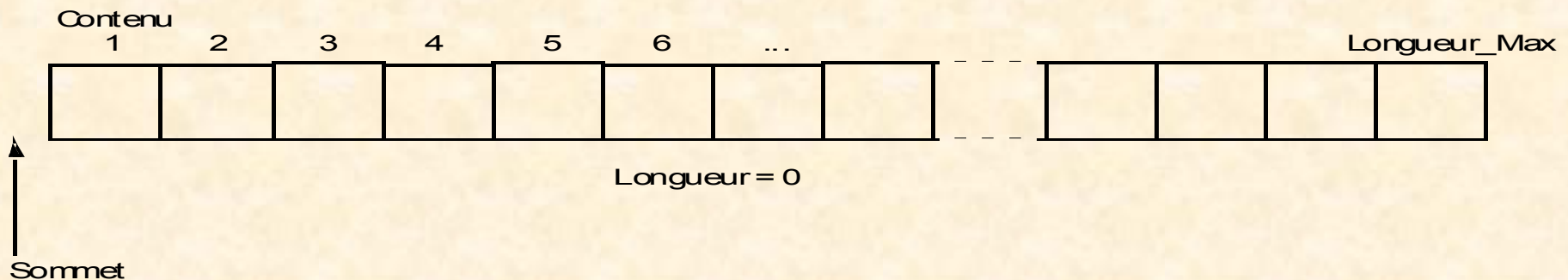
- Piles statiques

```

Longueur_Max : constant := 100;  -- Longueur maximum d'une pile
subtype T_Longueur is Integer range 0..Longueur_Max;
subtype T_Numerotation is Integer range 1..Longueur_Max;
subtype T_Pos_Sommet is Integer range 0..Longueur_Max;
type T_Contenu is array ( T_Numerotation ) of T_Info;          --*
type T_Pile_Statique is      -- Pour une pile statique
  record
    -- Contenu de la pile
    Contenu : T_Contenu ;          --*
    Longueur : T_Longueur := T_Longueur'First;  -- Longueur de la pile
    Sommet : T_Pos_Sommet := T_Pos_Sommet'First; -- Sommet de la pile
  end record;

```

Pile statique





-- Bases de gestion de piles statiques

**package** Piles\_Statiques **is**

**type** T\_Info **is** ...;-- Depend de l'application

Longueur\_Max : **constant** := 100;-- Longueur maximum d'une pile

**subtype** T\_Longueur **is** Integer **range** 0..Longueur\_Max;

**subtype** T\_Numerotation **is** Integer **range** 1..Longueur\_Max;

**subtype** T\_Pos\_Sommet **is** Integer **range** 0..Longueur\_Max;

**type** T\_Contenu **is** array ( T\_Numerotation ) **of** T\_Info;     --\*

**type** T\_Pile\_Statique **is**     -- Pour une pile statique

**record**

    -- Contenu de la pile

    Contenu : T\_Contenu;                                     --\*

    Longueur : T\_Longueur := T\_Longueur'First;     -- Longueur de

    Sommet : T\_Pos\_Sommet := T\_Pos\_Sommet'First; -- Sommet de

**end record**;

Pile\_Vide : **exception**;             -- Levee si la pile est vide

Pile\_Pleine : **exception**;         -- Levee si la pile est pleine

-----  
-- Insere Info au sommet de Pile. Leve Pile\_Pleine si l'insertion est  
-- impossible (la pile contient deja Longueur\_Max elements)

**procedure** Empiler (Pile : **in out** T\_Pile\_Statique; Info : **in** T\_Info );

-- Supprime l'information (au sommet de Pile) qui est rendue dans Info.

-- Leve Pile\_Vide si la suppression est impossible (la pile est vide)

**procedure** Desempiler ( Pile : **in out** T\_Pile\_Statique; Info : **out** T\_Info );

-----  
-- Change l'information du sommet de Pile. Leve Pile\_Vide si la

-- modification est impossible (la pile est vide)

**procedure** Modifier (Pile : **in out** T\_Pile\_Statique; Info : **in** T\_Info );

-----  
-- Retourne l'information du sommet de Pile. Leve Pile\_Vide si la

-- consultation est impossible (la pile est vide)

**function** Sommet ( Pile : T\_Pile\_Statique) **return** T\_Info;

-----  
-- Retourne True si Pile est vide, False sinon

**function** Vide ( Pile : T\_Pile\_Statique) **return** Boolean;

-----  
-- Retourne True si l'information Info est presente dans la pile,

**function** Recherche (Pile : T\_Pile\_Statique; Info : T\_Info) **return** Boolean;

-----  
-- Effectue un traitement sur tous les elements de la pile

**procedure** Parcourir ( Pile : **in out** T\_Pile\_Statique );

**end** Piles\_Statiques;

- **Piles dynamiques**

Pile dynamique

Sommet



```
type T_Element;                -- Predeclaration

-- Le type d'une pile dynamique
type T_Pile_Dynamique is access T_Element;

type T_Element is            -- Pour une variable dynamique
record
    Information : T_Info;      -- L'information
    Suivant : T_Pile_Dynamique; -- Pour reperer l'element qui suit
end record;
```

```

-- Bases de gestion de piles dynamiques
package Piles_Dynamiques is
  type T_Info is ...;           -- Depend de l'application
  type T_Element;               -- Predeclaration
  type T_Lien is access T_Element; -- Le type acces
  type T_Element is           -- Pour une variable dynamique
    record
      Information : T_Info;      -- L'information
      Suivant : T_Lien;         -- Pour l'element qui suit
    end record;
  type T_Pile_Dynamique is     -- Pour pile dynamique
    record
      Sommet : T_Lien;          -- Sommet de la pile
    end record;
  Pile_Vide : exception;      -- Levee si pile est vide
  -----
  -- Retourne True si Pile est vide, False sinon
  function Vide ( Pile : T_Pile_Dynamique) return Boolean;

```

-- Insere Info au sommet de Pile

```
procedure Empiler ( Pile : in out T_Pile_Dynamique; Info : in T_Info );
```

-- Supprime l'information rendue par Info.

```
procedure Desempiler(Pile:in out T_Pile_Dynamique; Info : out T_Info );
```

-- Change l'information du sommet de Pile.

-- Modification impossible si la pile est vide

```
procedure Modifier ( Pile : in T_Pile_Dynamique; Info : in T_Info );
```

-- Retourne l'information du sommet de Pile.

```
function Sommet ( Pile : T_Pile_Dynamique) return T_Info;
```

-- Retourne True si l'information Info presente

```
function Recherche (Pile : T_Pile_Dynamique; Info : T_Info)  
    return Boolean;
```

-- Effectue un traitement sur tous les elements

```
procedure Parcourir (Pile : in T_Pile_Dynamique);
```

```
end Piles_Dynamiques;
```

**Empiler**      **Desempiler**

