Chapitre 3.
Les tableaux: Algorithmes de Recherche et de Tri

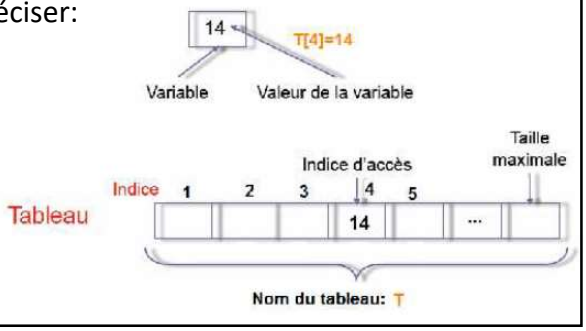
Définitions :

- Un **tableau** est une **variable structurée**, composé d'un **ensemble d'éléments** de **même type**, désigné par un **nom unique (identificateur).**
- Le **type** d'un tableau précise le type (**commun**) de **tous les éléments**
- L'**ensemble des éléments** d'un tableau sont, **ordonnés** (*cases mémoires numérotées*), identifiés par un **nom** et directement **accessibles** au moyen d'un **indice**.
- Un **indice**, est une **variable entière**, permet d'indiquer la **position d'un élément** donné au sein du tableau et de déterminer **sa valeur.**

Pour définir une **variable** de type **tableau**, il faut préciser:

- Le **nom** pour identifier le **tableau**
- Le **type des éléments** (*entier, réel, caractère, etc*)
- **L'indice**

Chaque **variable** du tableau est donc **caractérisée** par le **nom du tableau** et **son indice**.



Déclaration d'un tableau

La **déclaration d'un tableau** à **une dimension** montre en particulier sa **taille** et le **type** de ces éléments.

Syntaxe :

Var Nom_Tableau: tableau [borne_inf ... borne_sup] de type_éléments

OU

Var Nom_Tableau: tableau [borne_sup] de type_éléments

Le **tableau** contient (**borne_sup - borne_inf + 1**) éléments

Exemples :

Var T: tableau[1..20] d'entier — *On déclare un tableau qui stockera 20 valeurs entières*

OU

Var T: tableau[20] d'entier

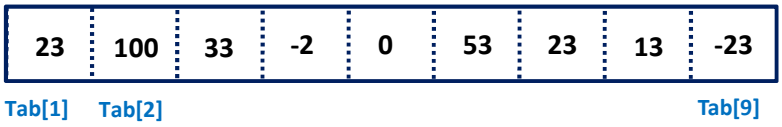
Var T2:tableau[1..20] de réel — *On déclare un tableau qui stockera 20 valeurs réelles*

Var T3:tableau[1..30] de caractère — *On déclare un tableau qui stockera 30 caractères*

Identification d'un élément du Tableau

Les tableaux à une dimension ou vecteurs

Var Tab : tableau[9] d'entier;



Ce tableau est de longueur 9, car il contient 9 emplacements.

Chacun des neufs valeurs du tableau est repéré par **son indice.**

Pour accéder à un élément du tableau, il suffit de préciser entre **crochets l'indice de la case contenant cet élément:**

- Pour **accéder au 6ème élément (53)**, on écrit : **Tab[6]**
- **Tab[3]** désigne la valeur du **3ème élément,**

D'une façon générale **Tab[i]** désigne le **ième élément** du tableau **Tab**.

Affectation d'un tableau à une dimension

L'affectation peut se faire de deux façon, soit:

1. avec des **valeurs initiales**
2. avec des **valeurs entrées au clavier**

Exemple 1 : Affectation avec des valeurs initiales

```
Algorithme affectation1
Var
      T: tableau[1..7] d Entier
Début
   T[1] ← 1
   T[2] ←5
   T[3] ← 7
   // les composantes T[4] à T[7] sont calculées
   à partir de T[1] à T[3]
   T[4] ← T[1]+T[2]
   T[5] ← T[3]-T[2]
   T[6] ← T[1]*T[3]
   T[7] ← T[1]+T[2]+T[3]
Fin
```

Affectation d'un tableau à une dimension

L'affectation peut se faire de deux façon, soit:

1. avec des **valeurs initiales**
2. avec des **valeurs entrées au clavier**

Exemple 2 : Affectation avec des valeurs entrées au clavier

```
Algorithme affectation2
Var
      T: tableau[7] de Entier
      i : entier
Début
/* lecture des éléments du tableau */
Pour i ← 1 à 7 Faire
   Ecrire("entrer l'élément N° ", i)
   Lire(T[i])
FinPour
Fin
```

Exercice 1: Calculer la somme, le produit et la moyenne

Ecrire un **algorithme** permettant d'entrer **N** **valeurs réelles** au clavier, les **stocker dans un tableau**, calculer **leur somme**, **leur produit** et **leur moyenne.**

```
Algorithme tableau_somme
Var   T: tableau[100] de réels
      Som, prod, Moy: réel
      i,N : entier
Début
   Ecrire("saisir le nombre d'éléments du tableau")
   Lire(N)

   // lecture des éléments du tableau
   Pour i  ← 1 à N  faire
     Ecrire("entrer l'élément numéro ", i)
     Lire(T[i])
   FinPour
```

```
// affichage des éléments du tableau
Pour i ← 1 à N faire
    Ecrire (" l'élément numéro ",i, " est : ", T[i])
   // OU  Ecrire (T[i])
Finpour
Fin
// calcul de la somme,produit et moy d'éléments du tableau
Som ← 0
Prod ← 1
    Pour i ← 1 à N faire
          Som ← Som+ T[i]
          prod ←  prod* T[i]
    FinPour
Moy ← Som/N
Ecrire("la somme et le produit des éléments du tableau",Som,prod)
Ecrire("la moyenne  des éléments du tableau" ,Moy)
Fin
```

Exemple 2 : Consultation d'un élément dans un tableau

Ecrire un algorithme permettant d'entrer **N** valeurs réelles au clavier (*avec un control de saisie sur N*), les stocker dans un tableau et afficher un élément connaissant son indice.

```
Algorithme consultation
Var
        T: tableau[1..100] de réels
        N,p,i: entier
Début
répéter
   Ecrire("saisir le nombre d'éléments du tableau")
   Lire(N)
Jusqu'à (N>=0 ET N=<100)
Si (N=0) alors
   Ecrire("le tableau est vide")
Sinon
```

```
        // lecture des éléments du tableau
        Pour i ←1 à N Faire
          Ecrire("entrer l'élément N° ", i)
          Lire(T[i])
        FinPour i
        // affichage des éléments du tableau
        Pour i ← 1 à N faire
            Ecrire (T[i])
        FinPour i
            Ecrire("entrer l'indice de l'élément à consulter")
            Lire(p)
        Si (p<1) OU (p>N) alors
                Ecrire("position hors limites du tableau ")
            Sinon
                Ecrire("l'élément à consulter est ", T[p])
            Finsi
        Finsi
Fin
```

Autres applications :

1. **Recherche du plus grand élément dans un tableau :**

   On note **Max**, le plus grand élément du tableau **T** et **p** son indice

2. **Recherche d'un élément dans un tableau connaissant sa valeur (toutes les occurrences) :**

   L'élément recherché peut apparaitre une ou plusieurs fois dans le tableau, dans ces cas on va afficher ses positions. Mais il peut ne pas apparaitre, c'est pourquoi on va introduire une variable **booléenne « existe »** (ou un entier **k** qui prend la valeur **0** ou **1**) qui va marquer l'existence ou non de l'élément recherché dans le tableau.

3. **Modification d'un élément dans un tableau connaissant sa position :**

   Cet algorithme permet de modifier, la valeur d'un élément connaissant sa position dans le tableau.

Les variables dimensionnés (les tableaux)

1. **Recherche du plus grand élément dans un tableau**

On note **Max**, le **plus grand élément** du tableau **T** et **p** son indice, l'algorithme est :

```
Algorithme maximum
  Varibales
      T : tableau[100] de réels
      N, p, i : entier
      Max : réel
  Début
      Répéter
            Ecrire("saisir le nombre d'éléments du tableau")
            Lire(N)
      Jusqu'à (N>=0 ET N<=100)
      Si (N=0) alors
            Ecrire("le tableau est vide")
      Sinon
```

```
// Lecture des éléments du tableau
Pour i ← 1 à N Faire
        Ecrire("entrer l'élément N° ", i)
        Lire(T[i])
FinPour

//Affichage des éléments du tableau
Pour i ← 1 à N faire
        Ecrire (T[i])
FinPour

// Recherche et affichage de max
        Max ← T[1]
        p ← 1
Pour i ← 2 à N faire
        Si(T[i]>Max) alors
                Max ← T[i]
                p ← i
        Finsi
FinPour
        Ecrire("Le plus grand élément du tableau est ",Max, ",se trouve à la position ",p)
 Finsi
Fin
```

Les variables dimensionnés (les tableaux)

2. Recherche d'un élément dans un tableau connaissant sa valeur (toutes les occurrences)

L'élément recherché peut apparaitre une ou plusieurs fois dans le tableau, dans ces cas on va afficher ses positions. Mais, il peut ne pas apparaitre, c'est pourquoi on va introduire une variable booléenne « existe » (ou un entier **k** qui prend la valeur **0** ou **1**) qui va marquer l'existence ou non de l'élément recherché dans le tableau. ***x : étant la valeur de l'élément à chercher***

```
Algorithme rechercheTTesOccurences
    Variables
       T : tableau[1..100] de réels
       N, i, K : entier
       x : réel
     Début
       Répéter
              Ecrire("Saisir le nombre d'éléments du tableau")
              Lire(N)
       Jusqu'à (N>=0 ET N<=100)
       Si (N=0) alors
              Ecrire("Le tableau est vide")
       Sinon
```

```
Pour i←1 à N Faire
        Ecrire("Entrer l'élément N° ", i)
        Lire(T[i])
FinPour
// affichage des éléments du tableau
Pour i← 1 à N faire
        Ecrire (T[i])
FinPour
Ecrire("Entrer la valeur de l'élément à chercher ")
Lire (x)
K← 0
C←0
Pour i← 1 à N faire
        Si(T[i]=x)  alors
                K← 1
                C←C+1
                Ecrire("L'élément à chercher apparait à la position : ", i)
        Finsi
FinPour
Si (K=0) alors
        Ecrire("L'élément à chercher n'apparait pas dans ce tableau ")
Sinon
        Ecrire("L'élément à chercher apparait dans ce tableau ", C, " fois")
```

3. Modification d'un élément dans un tableau connaissant sa position

Cet algorithme permet de modifier, la valeur d'un élément connaissant sa position dans le tableau.

```
Algorithme Modification
  Variables
        T : tableau[100] de réels
        N, p : entier
        x : réel
  Début
        répéter
            Ecrire(''saisir la dimension du T'')
            Lire(N)
        Jusqu'à (N>=0 ET N<=100)
        Si (N=0) alors
             Ecrire('' le tableau est vide'')
        Sinon
        //lecture des éléments du tableau
            Pour i ← 1 à N Faire
                Ecrire(" T[",i, "]= ")
                Lire(T[i])
            FinPour i
    //affichage des éléments du tableau
    Pour i ← 1 à N faire
          Ecrire (T[i])
    FinPour i
    Ecrire("Entrer l'indice de l'élément à modifier ")
    Lire (p)
    Si ((p<1) OU (p>N)) alors
        Ecrire(" Position hors limites du tableau ")
    Sinon
        Ecrire("L'ancienne valeur est : ", T[p])
        Ecrire("Entrer la nouvelle valeur: ")
        Lire(x)
        T[p] ← x
    Finsi
    //affichage des éléments après modification
    Pour i ← 1 à N faire
          Ecrire (T[i])
    FinPour i
  Finsi
Fin
```

Algorithmes de Tri

- Un **algorithme de tri** est un algorithme qui permet **d'organiser une collection d'objets (tableau, liste, …)** selon un **ordre déterminé (croissant ou décroissant)**

50	12	86	3	954	20	124

  - **Tri croissant** : si l'élément d'indice i est inférieur ou égal à l'élément d'indice i+1.

3	12	20	50	86	124	954

  - **Tri décroissant** : si l'élément d'indice i est supérieur ou égal à l'élément d'indice i+1.

954	124	86	50	20	12	3

- Il existe **plusieurs algorithmes** permettant de trier un tableau:
  - Tri par sélection
  - Tri à bulle
  - …
  - Tri par extraction
  - Tri rapide
  - Tri par insertion
  - Tri par fusion

➔ Les **algorithmes de tri** ont tous leurs points forts et leurs points faibles : **algorithme lent** ou **rapide**, **gourmant en mémoire**, **complexe**, ...

Tri par sélection d'un tableau

Principe:

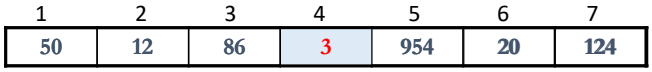
1. Rechercher le plus petit élément du tableau et l'échanger avec le premier élément
2. Rechercher le plus petit élément entre les **positions 2 et n** et l'échanger avec le deuxième élément
3. Rechercher le plus petit élément entre les **positions 3 et n** et l'échanger avec le troisième élément
4. …

Soit le tableau suivant:

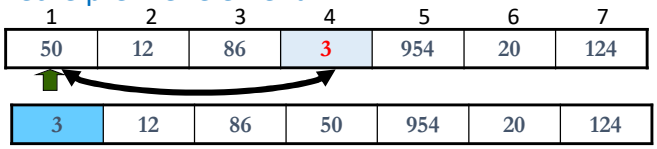
1	2	3	4	5	6	7
50	12	86	3	954	20	124

Tri par sélection d'un tableau

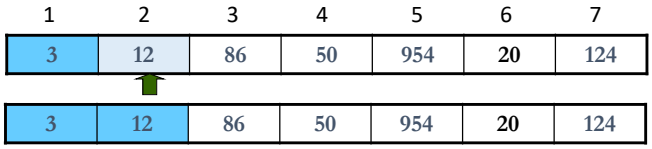
1. **Rechercher** le plus petit élément du tableau



et **l'échanger** avec le premier élément

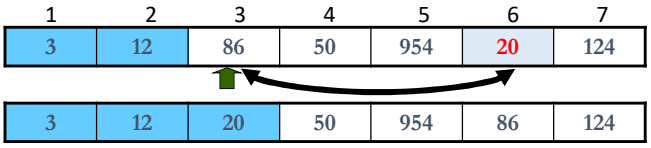


2. Rechercher le plus petit élément entre les positions 2 et n(7) et l'échanger avec le deuxième élément

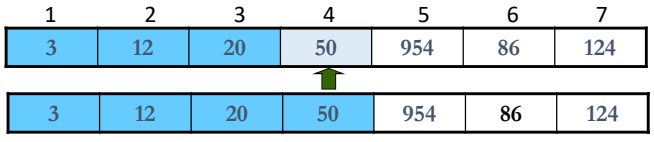


Tri par sélection d'un tableau

3. Rechercher le plus petit élément entre les positions 3 et n(7) et l'échanger avec le troisième élément



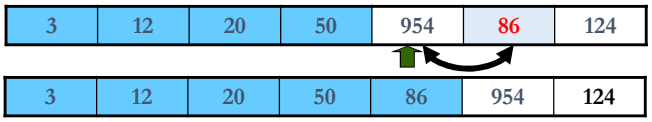
4. Rechercher le plus petit élément entre les positions 4 et n(7) et l'échanger avec le quatrième élément



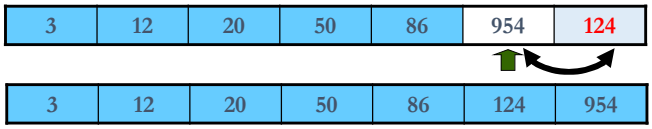
5. Rechercher le plus petit élément entre les positions 5 et n(7) et l'échanger avec le cinquième élément

Tri par sélection d'un tableau

5. Rechercher le plus petit élément entre les positions 5 et n(7) et l'échanger avec le cinquième élément



6. Rechercher le plus petit élément entre les positions 6 et n(7) et l'échanger avec le sixième élément



Le tableau est maintenant trié dans un ordre croissant

De même, pour trier un tableau dans **ordre décroissant** en utilisant le **tri par sélection,** il faut rechercher le plus grand élément du tableau et l'échanger avec le premier élément ….

Algorithme du tri par sélection d'un tableau