Chapitre 3

Algorithmes de Recherche et de Tri des tableaux

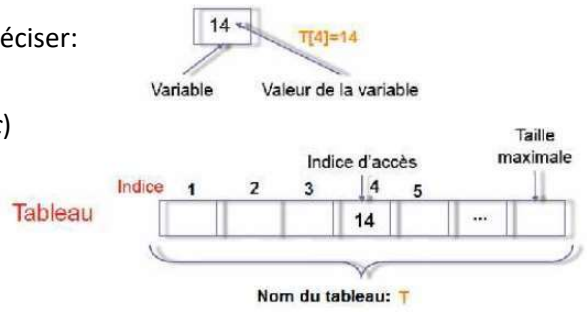
Définitions : Rappel

- 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 indique le type de **tous les éléments**
- L'**ensemble des éléments** d'un tableau sont, **ordonnés** (*cases numérotées*), identifiés par un **nom** et directement **accessibles** au moyen d'un **indice**.
- L'**indice**, est une **variable entière**, permettant d'indiquer la **position d'un élément** donné du tableau afin de déterminer **sa valeur.**

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

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

Chaque **élément** du tableau est **caractérisé** par le **nom du tableau** et l'**indice de l'élément.**



Déclaration d'un tableau : Rappel

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

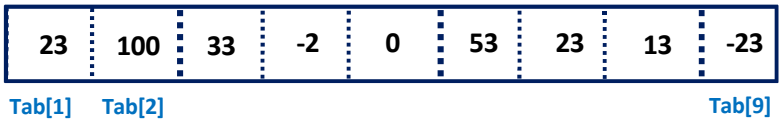
Var T2:tableau[1..20] de réel — *On déclare un tableau qui stockera 20 valeurs réelles*

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

Identification d'un élément du Tableau : Rappel

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 **neuf 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 : Tab[i]** désigne le **$i^{ème}$ élément** du tableau **Tab**.

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

Affectation d'un tableau à une dimension : Rappel

L'affectation peut se faire de deux façons, 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 : Rappel

L'affectation peut se faire de deux façons, 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
```

Exercice 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.

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**, …

17

Tri par sélection d'un tableau

Principe:

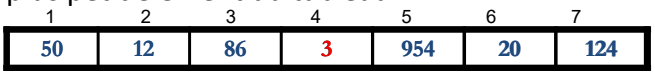
1. Rechercher le plus petit élément du tableau et l'échanger avec le premier élément
2. Rechercher le plus petit élément entre les positions 2 et n et l'échanger avec le deuxième élément
3. Rechercher le plus petit élément entre les positions 3 et n et l'échanger avec le troisième élément
4. ...

Soit le tableau suivant:

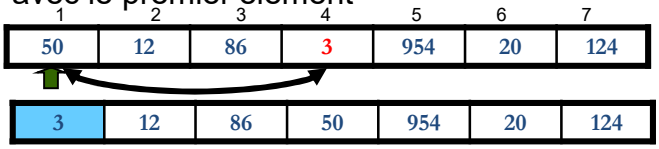
50	12	86	3	954	20	124
1	2	3	4	5	6	7

Tri par sélection d'un tableau

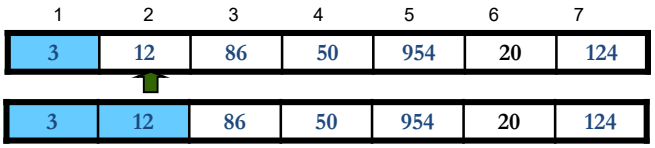
1. Rechercher le plus petit élément du tableau



et l'échanger avec le premier élément

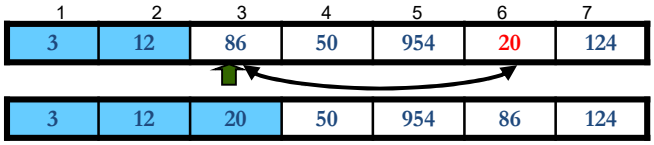


2. Rechercher le plus petit élément entre les positions 2 et 7 et l'échanger avec le deuxième élément

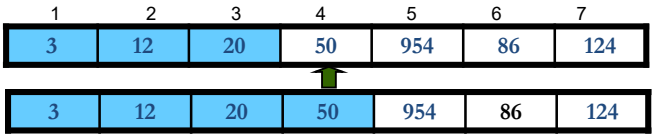


Tri par sélection d'un tableau

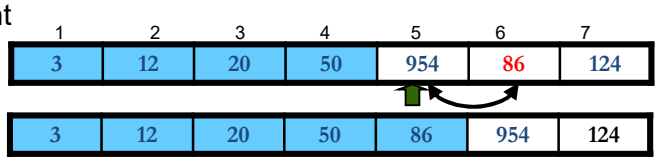
3. Rechercher le plus petit élément entre les positions 3 et 7 et l'échanger avec le troisième élément



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

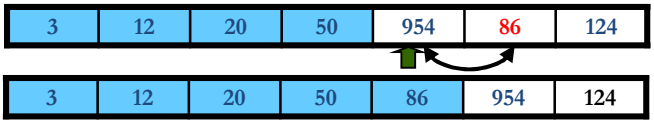


5. Rechercher le plus petit élément entre les positions 5 et 7 et l'échanger avec le cinquième élément

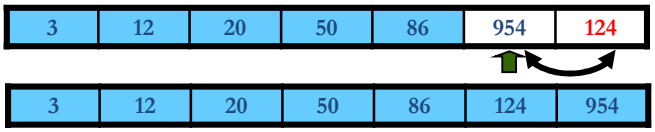


Tri par sélection d'un tableau

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



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



Le tableau est maintenant trié dans un ordre croissant

Tri par sélection d'un tableau

L'algorithme est:

```
Pour i allant de 1 à N-1 faire
        indice_min ← i
        Pour j allant de i+1 à N faire  //on va chercher la valeur minimale
                Si (T[j] < T[indice_min])   alors
                        indice_min ← j
                Finsi
        Finpour j
        Si ( i ≠ indice_min) alors          //on a trouvé une valeur minimale ayant indice_min
                Aide ← T[indice_min]
                T[indice_min] ← T[i]
                T[i] ← Aide
        Finsi
Finpour i
```

Tri par sélection d'un tableau

(programme python)

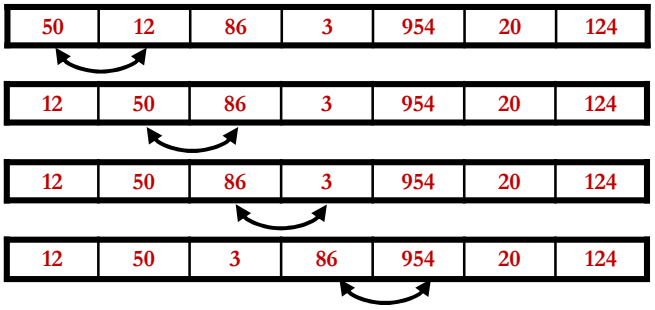
```
def TriparSelection (T, N) :
    for i in range(N-1):
        indice_min=i
        for j in range (i+1,N):
            if (T[j]<T[indice_min]):
                indice_min=j
        if (i!=indice_min):
            Aide = T[indice_min]
            T[indice_min]=T[i]
            T[i]=Aide


T=[14,3,8,5,14,25,18,2]
print(len(T))
print(T)
TriparSelection(T, len(T))
print(T)
```

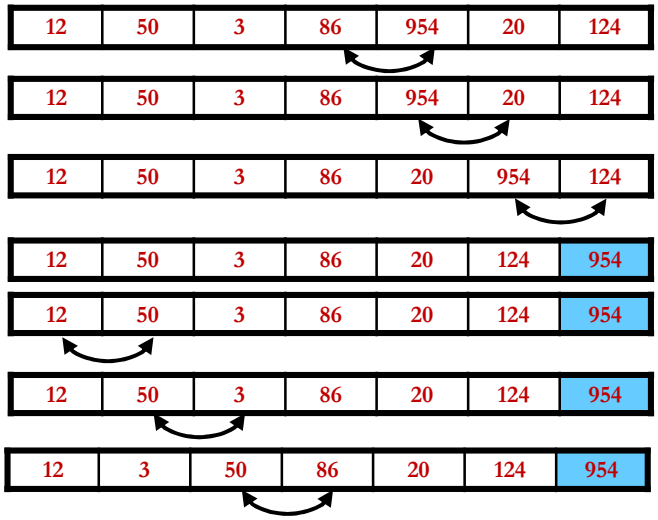
Tri à bulle d'un tableau

1. Parcourir tout le tableau en comparant successivement les éléments du tableau deux à deux. Permuter les deux éléments comparés s'ils ne sont pas dans l'ordre.
2. Cette opération est répétée plusieurs fois jusqu'à ce que le tableau soit entièrement parcouru sans réaliser aucune permutation.

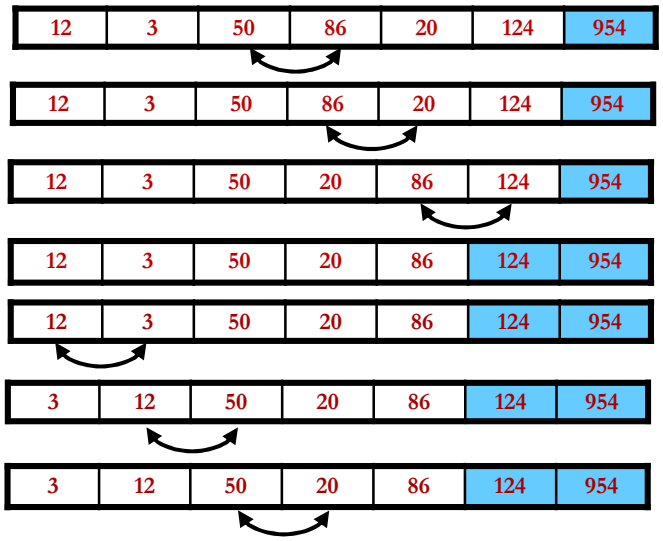
Soit le tableau suivant:



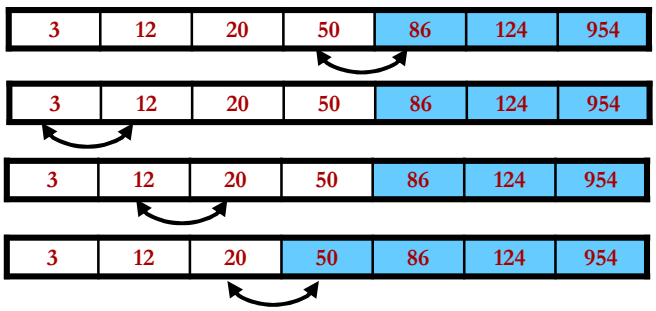
Tri à bulle d'un tableau



Tri à bulle d'un tableau



Tri à bulle d'un tableau



En parcourant tout le tableau (jusqu'au dernier élément) sans faire aucune permutation, cela signifie que le Tableau est dans l'ordre (le tableau est trié)

Tri à bulle d'un tableau

Version 1

```
Pour i allant de 1 à N-1 faire
        Pour j allant de 1 à N-1 faire
                Si (T[j] > T[j+1]) alors
                        Aide ← T[j]
                        T[j] ← T[j+1]
                        T[j+1]← Aide
                Finsi
```

Version 2

```
Pour i allant de 1 à N-1 faire
        Pour j allant de 1 à N-i-1 faire
                Si (T[j] > T[j+1]) alors
                        Aide ← T[j]
                        T[j] ← T[j+1]
                        T[j+1]← Aide
                Finsi
```


Tri à bulle d'un tableau

Version 3

```
terme_tri ← N-1
Répéter
        ordre ← vrai;
        Pour i allant de 1 à terme_tri faire
                Si (T[i] > T[i+1]) alors
                        Aide ← T[i]
                        T[i] ← T[i+1]
                        T[i+1]← Aide
                        ordre ← faux
                Finsi
        Finpour
        Si (ordre = faux) alors
                terme_tri ← terme_tri -1
        Finsi
Jusqu'à (ordre = vrai)
```

Tri à Bulle (programme python)

```
1   #programme Python pour l'implémentation du Tri à Bulle
2   def tri_bulle(tab):
3       n=len(tab)
4       #parcourir tous les éléments du tableau
5       for i in range(n):
6           for j in range (n-1):
7               #échanger si l'élément trouvé est plus grand que le suivant
8               if tab[j]>tab[j+1]:
9                   tab[j], tab[j+1]=tab[j+1], tab[j]
10
11
12  #programme principal pour tester le code ci-dessus
13  tab=[98,22,15,32,2,74,63,70]
14  tri_bulle(tab)
15  print("Le tableau trié est:")
16  print(tab)
17
```

Tri par extraction

Cette méthode utilise en plus du tableau à trier un deuxième tableau dans lequel on place les éléments triés.

- On cherche le plus petit élément min dans le premier tableau T1 et on le place au début du deuxième tableau T2.
- Ensuite on cherche le plus petit élément parmi ceux non encore sélectionnée du T1 et on le place dans T2 jusqu'à ce que tous les éléments soient recopiés dans T2.
- A chaque fois qu'un élément est sélectionné, il est remplacé par une valeur spéciale pour ne pas être sélectionné une deuxième fois.

Exemple: soit un tableau T1 suivant contient 6 entiers:

T1	8	4	15	6	3	11
Tableau résultat T2	3	4	6	8	11	15

Tri par extraction

```
Algorithme triParExtraction;
 Var
      T1,T2: tableau[1..100] de réel;
      i, j, max, N, indice : entier;
 Début
                  // Saisie des éléments du tableau
                  Pour i ← 1 à N faire
                                    Ecrire("entrer l'élément T1 ",i);
                                    Lire(T1[i]);
                  FinPour
                  // Affichage  des éléments du tableau avant le tri
                  Pour i ← 1 à N faire
                                    Ecrire(T1[i]);
                  FinPour
                  // Valeur spéciale pour remplacer l'élément sélectionné
                  max=1000
```

```
        // Trier le tableau à l'aide de l'algorithme de tri par extraction
          Pour i ← 1 à N faire
                    indice ← 1;
               Pour j ← 2 à N faire
                    Si(T1[indice]>T1[j]) alors
                          indice ← j;
                    Finsi
               FinPour j
                     T2[i] ← T1[indice];
                     T1[indice] ← max;
          FinPour i
        // Affichage  des éléments du tableau trié
         Pour i ← 1 à N faire
              Ecrire(T2[i]);
        FinPour i

Fin
```

```
#programme Python pour l'implémentation du Tri par extraction

tab=[98,22,15,32,2,74,63,70]
print("Le tableau avant le tri est: ")
print(tab)
tab2= [None] * len(tab)
n=len(tab)
max=10000
#parcourir tous les éléments du tableau
for i in range(n):
        indice=0
        # On cherche le plus petit élément dans le premier tableau
        for j in range (1,n):
            if (tab[indice]>tab[j]):
                indice = j
        # On le place le plus petit élément au début du deuxième tableau
        tab2[i]=tab[indice]
        tab[indice]=max
print("Le tableau après le tri par extraction est: ")
print(tab2)
```