Chapitre 4.
Les variables dimensionnées
(les tableaux)

Notion de variables structurées

Jusqu'à présent, on a utilisé des **variables** de **type standards** (**variable simple**) et qui prennent une **seule valeur et un seul emplacement** mémoire.

Dans de nombreuse applications, on souhaite **regrouper** plusieurs valeurs dans **une seule variable** (**variable structurée**).

Exemple 1 : Supposons que nous souhaitons déterminer à partir de **30 notes** fournies en entrée, le **nombre d'étudiants** qui ont une **note supérieure à la moyenne** de la classe.

Pour parvenir à un tel **résultat**, nous devons :

1. Lire les **30 notes**,
2. Déterminer la **moyenne** de la classe,
3. Compter combien de notes parmi les **30 notes** qui sont supérieures à la moyenne,

➔ Il faut **conserver les notes** en mémoire afin qu'elles soient accessibles durant l'**exécution du programme.**

Solution 1 (Avec des variables simples):

Utiliser **30 variables réelles** nommées **N1, N2, …, N30** pour conserver les valeurs des 30 notes.
Mais cette façon de faire présente des inconvénients :

1. Il faut **réserver un nom de variable par note** ;
2. Il n y a **aucun lien entre ces variables**. Or, dans certains cas, on doit appliquer le même traitement sur l'ensemble des variables:

Algorithme très lourd à écrire

```
// Déterminer les notes > à la moyenne
Nbre ← 0;
Si (N1  >Moy_classe) alors
        Nbre ←Nbre+1;
FinSi
        …
Si (N30> Moy_classe) alors
        Nbre ←Nbre+1;
FinSi
```

Afin d'éviter la **multiplication du nombre des variables (30)**, les langages de programmation offrent la possibilité de rassembler toutes ces variables dans une seule **structure de données** appelée : **Tableau**

Solution 2 (Avec un tableau) :

Regrouper toutes ces notes (N1, N2, … N30) dans une seule variable structurée composée de 30 cases juxtaposées de **même nom** et de **même type** appelée **Tableau**.

1. attribuer un **seul nom** à l'ensemble des **30 notes**, par exemple **N**.
2. **repérer chaque note** par ce nom (**N**) suivi entre crochets d'un **numéro entre 1 et 30** : **N[1], N[2], … N[30]**

Ce **tableau** peut être présenter par le schéma suivant :

N[1]	N[2]	N[3]	N[4]				N[28]	N[29]	N[30]
15	13.5	14	10.25	…	…	…	11	18	19

Parfois, nous choisirons la **valeur 0** pour la **borne inférieure** dans le but de faciliter la **traduction des algorithmes vers d'autres langages de programmation** (C, Java, ...). Ainsi, on peut écrire:

N[0]	N[1]	N[2]	N[3]	…			N[27]	N[28]	N[29]
15	13.5	14	10.25	…	…	…	11	18	19

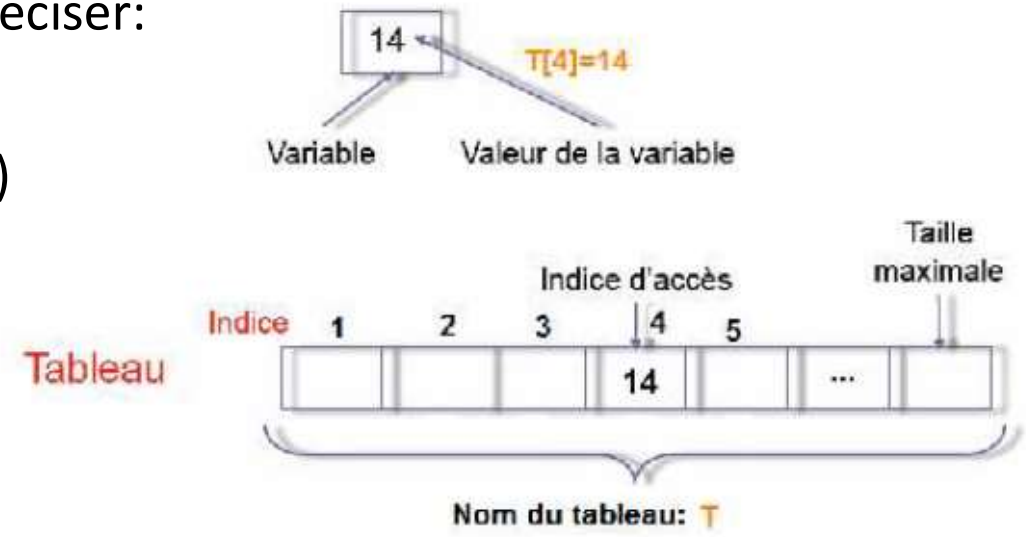
Définitions :

- 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;

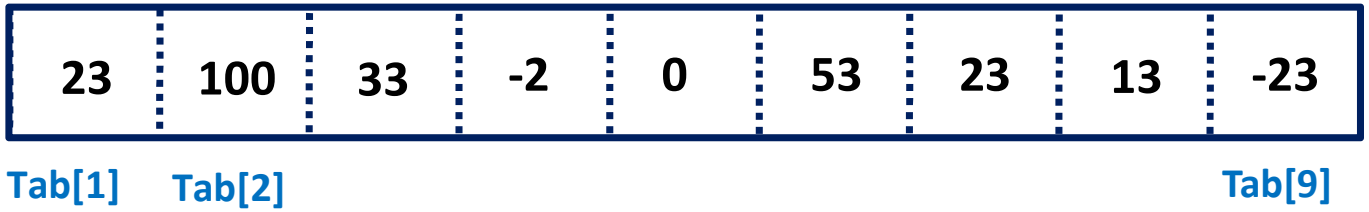
Var T2:tableau[1..20] de réel; *On déclare un tableau qui stockera 20 valeurs réelles*

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
```

Affectation et affichage d'un tableau à une dimension

Exercice 1:

Ecrire un **algorithme permettant d'entrer 10 valeurs réelles** au clavier, les **stocker dans un tableau et les afficher à l'écran.**

Solution :

On va utiliser un tableau **T** de **10** cases de **type réels**, et un **compteur i** pour parcourir le tableau à l'aide d'une boucle.

Ecrire un algorithme permettant d'entrer 10 valeurs réelles au clavier, les stocker dans un tableau et les afficher à l'écran.

```
Algorithme affiche_tableau;
  Var
        T: tableau[1..10] de réels;
          i : entier;
  Début
  // lecture des éléments du tableau
  Pour i  ← 1 à 10  Faire
    Ecrire("entrer l'élément N° ", i);
    Lire(T[i]);
  Finpour
  // affichage des éléments du tableau
  Pour i ← 1 à 10 faire
      Ecrire (" l'élément numéro ",i, " est : ", T[i]);
     // OU  Ecrire (T[i]);
  Finpour
  Fin
```

Exercice 2 :

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);
   Pour i ← 1 à N faire
      Ecrire("entrer l'élément numéro ", i);
      Lire(T[i]);
   FinPour
   // calcul de la somme,produit et moy d'éléments du tableau
```

```
Pour i ← 1 à N faire
   Ecrire("entrer l'élément numéro ", i);
   Lire(T[i]);
FinPour
// 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 2bis

Ecrire un algorithme permettant d'entrer **N** valeurs réelles au clavier (*avec un control de saisie sur N*), les stocker dans un tableau, calculer leur somme, produit et leur moyenne.

```
Algorithme tableau_ Calcul;
Var   T: tableau[100] de réels;
      Som, prod, Moy: réel;
       i,N : entier;
Début
répéter
   Ecrire("saisir le nombre d'éléments du tableau");
   Lire(N);
Jusqu'à (N>=1  ET  N=<100) ;
```

```
  Pour i  ← 1 à N  faire
     Ecrire("entrer l'élément numéro ", i);
     Lire(T[i]);
  FinPour
// 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 2bis

Ecrire un algorithme permettant d'entrer **N** valeurs réelles au clavier (*avec un control de saisie sur N*), les stocker dans un tableau, calculer leur somme, produit et leur moyenne.

```
Algorithme tableau_Calcul2;
Var   T: tableau[100] de réels;
      Som, prod, Moy: réel;
       i,N : 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
```

```
  Pour i  ← 1 à N  faire
     Ecrire("entrer l'élément numéro ", i);
     Lire(T[i]);
  FinPour
// 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);
Finsi
Fin
```

Exemple 3 : Consultation d'un élément dans un tableau

Cet algorithme permet d'afficher un élément du tableau T de dimension N 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
```

Exemple 3 : Consultation d'un élément dans un tableau

Cet algorithme permet d'afficher un élément du tableau T de dimension N 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
répéter
    Ecrire("entrer l'indice de l'élément à consulter");
    Lire(p);
Jusqu'à ( (p>=1) ET (p<=N) ) ;
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;
  Var
      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 rechercheTTes_occurences;
    Var
       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
```

```
// 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
    Ecrire(" entrer la valeur de l'élément à chercher ");
    Lire (x);
    K← 0;
    Pour i← 1 à N faire
                Si(T[i]=x)  alors
                            K← 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 ");
    Finsi
Finsi
Fin
```

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;
  Var
        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 dans cette position est : ", T[p]);
  Ecrire(" entrer la nouvelle valeur: ");
  Lire(x);
  T[p] ← x;
Finsi
//affichage des éléments du tableau 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 par insertion**
  - **Tri rapide**
  - **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:

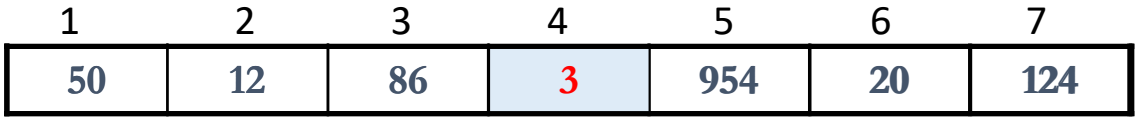
1. Rechercher le plus petit élément du tableau et l'échanger avec le premier élément
2. Rechercher le plus petit élément entre les **positions 2 et n** et l'échanger avec le deuxième élément
3. Rechercher le plus petit élément entre les **positions 3 et n** et l'échanger avec le troisième élément
4.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

Soit le tableau suivant:

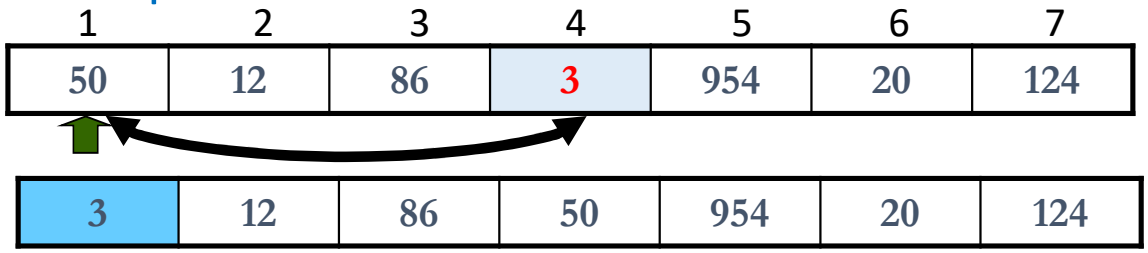
1	2	3	4	5	6	7
50	**12**	**86**	**3**	**954**	**20**	**124**

Tri par sélection d'un tableau

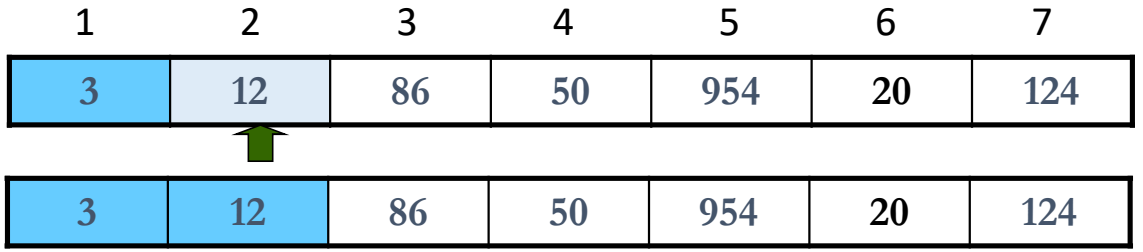
1. **Rechercher** le plus petit élément du tableau



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

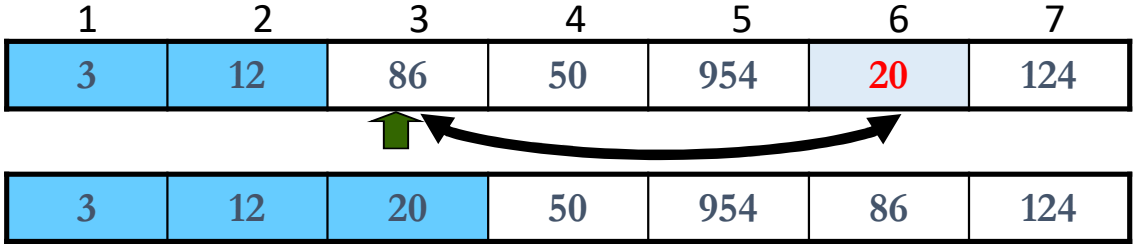


2. Rechercher le plus petit élément entre les positions 2 et n(7) et l'échanger avec le deuxième élément

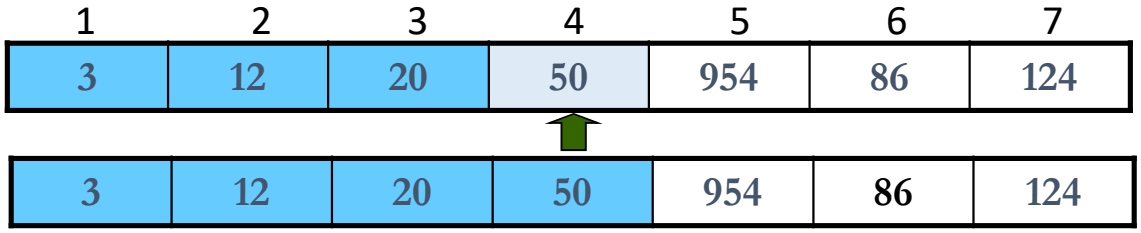


Tri par sélection d'un tableau

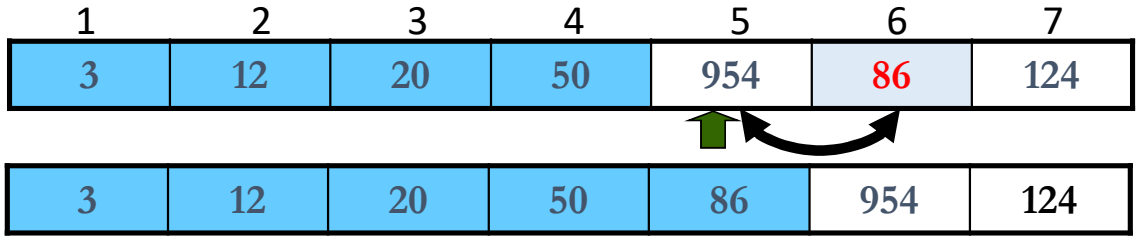
3. Rechercher le plus petit élément entre les positions 3 et n(7) et l'échanger avec le troisième élément



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

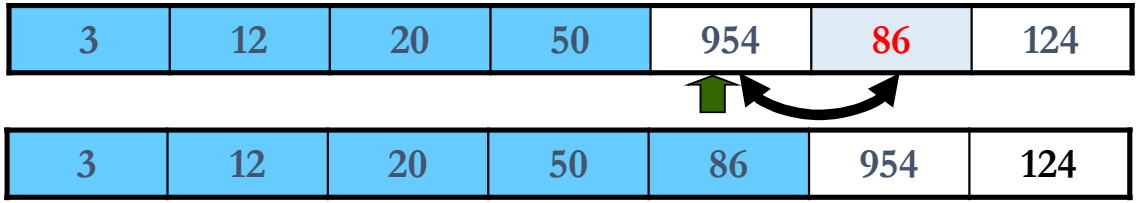


5. Rechercher le plus petit élément entre les positions 5 et n(7) et l'échanger avec le cinquième élément

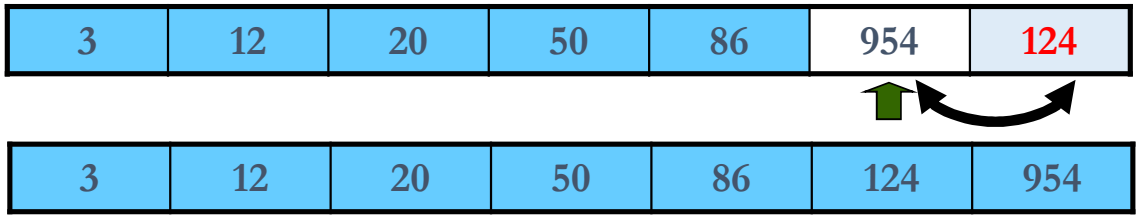


Tri par sélection d'un tableau

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

```
Pour i allant de 1 à N-1 faire
    indice_min ← i;
        Pour j ← i+1 à N faire      //Chercher la plus petite valeur dans le sous tableau de droite
            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 à bulle d'un tableau

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

Soit le **tableau suivant** :

Premier passage :

1. On se positionne dans la 1ere case :
   50>12 → On permute
2. On se positionne dans la 2ième case :
   50<86 → Pas de permutation
3. On se positionne dans la 3ième case :
   86>3 → On permute
3. On se positionne dans la 4ième case :
   86<954 → Pas de permutation

Tri à bulle d'un tableau

4. On se positionne dans la 4ième case :
   86<954 → Pas de permutation

5. On se positionne dans la 5ième case :
   954>20 → On permute

6. On se positionne dans la 6ième case :
   954>124 → On permute

Tri à bulle d'un tableau

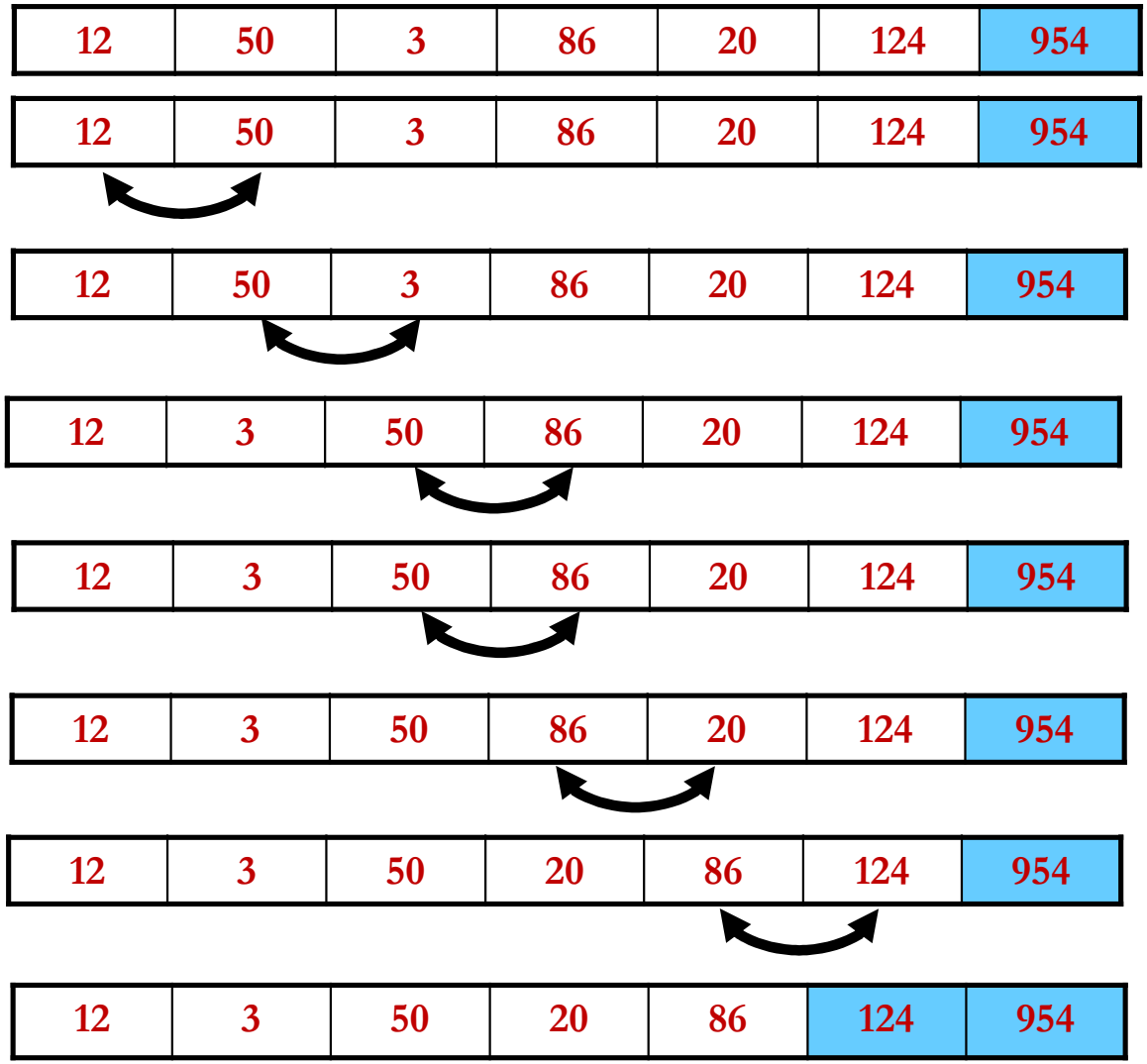
Remarques :

12	50	3	86	20	124	954

1. A l'issue de ce premier passage, on remarque que le **tableau n'est pas entièrement** trié, mais la **plus grande valeur** a été placée dans **la dernière case** du tableau (*colorée en bleu*).
2. Il faut **effectuer plusieurs passages** en vérifiant à chaque passage **si des permutations ont eu lieu.**
3. Quand une **permutation au moins a eu lieu** lors d'au passage, **il faut en relancer un autre**
4. Il faut **mettre en place un drapeau** indiquant si une permutation a eu lieu ou non

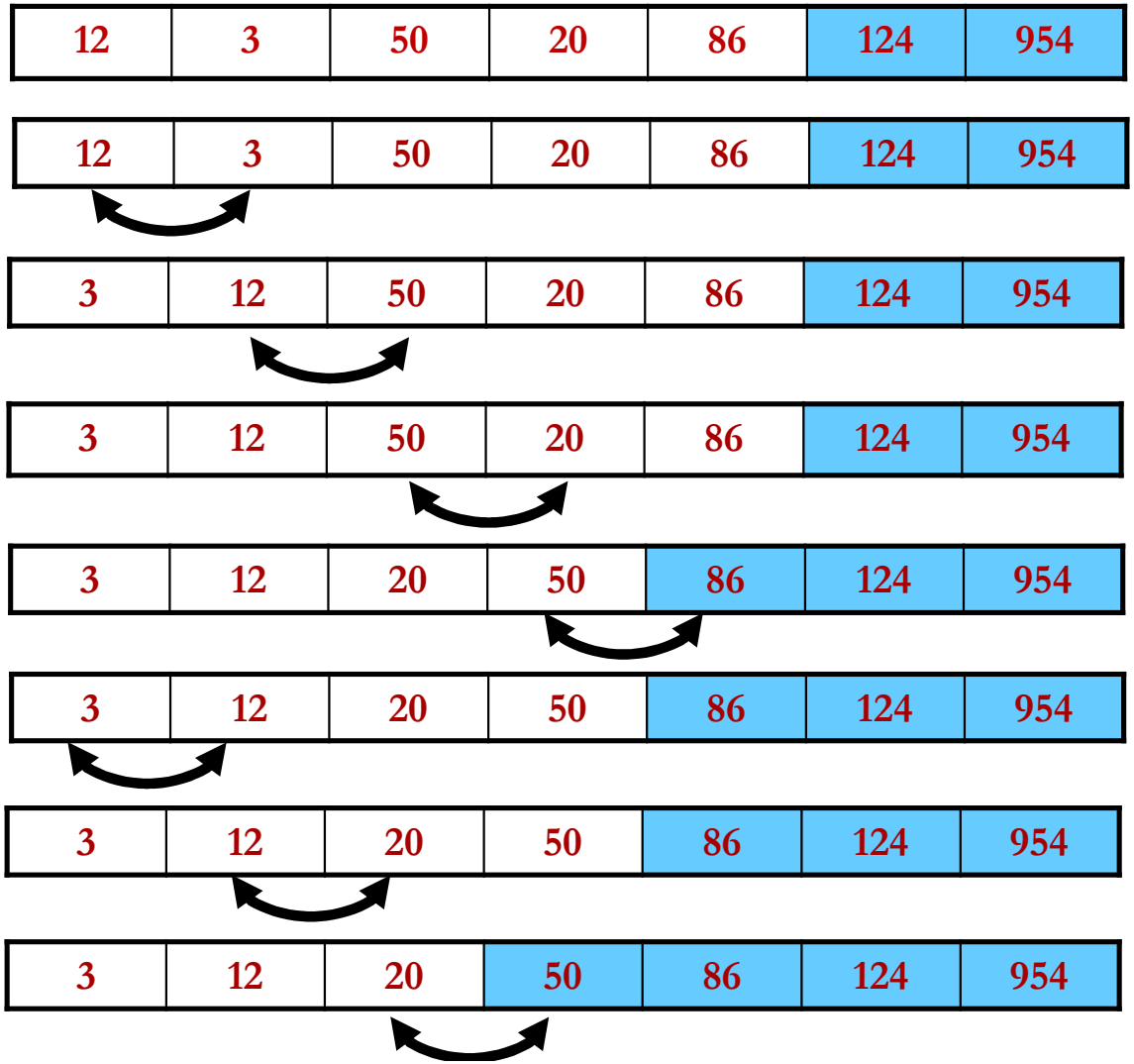
Tri à bulle d'un tableau

Deuxième passage :



Tri à bulle d'un tableau

Troisième passage :



On doit effectuer plusieurs passages … Jusqu'à ce que le tableau soit dans l'ordre (trié) : En parcourant tout le tableau sans faire aucune permutation.

Tri à bulle d'un tableau

```
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 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 (**notes d'étudiants, **valeur spéciale = 21)** :

T1	8	4	15	6	3	11
Tableau résultat T2						

Itération 1 :

T1	8	4	15	6	3	11
Tableau résultat T2	3					
T1 (après iteration 1)	8	4	15	6	21	11

Itération 2 :

T1	8	4	15	6	21	11
Tableau résultat T2	3	4				
T1 (après iteration 2)	8	21	15	6	21	11

Itération 3 :

T1	8	21	15	6	21	11
Tableau résultat T2	3	4	6			
T1 (après iteration 3)	8	21	15	21	21	11

Itération 4 :

T1	8	21	15	21	21	11
Tableau résultat T2	3	4	6	8		
T1 (après iteration 4)	21	21	15	21	21	11

Itération 5 :

T1	21	21	15	21	21	11
Tableau résultat T2	3	4	6	8	11	
T1 (après iteration 5)	21	21	15	21	21	21

Itération 6 :

T1	21	21	15	21	21	21
Tableau résultat T2	3	4	6	8	11	15
T1 (après iteration 6)	21	21	21	21	21	21

- Tous les éléments de T1 sont remplacés par une valeur spéciale (21).
- Tous les éléments sont triés et recopiés dans le tableau résultat T2.

Tri par extraction

```
Algorithme  triParExtraction;
  Var
        T1, T2 : tableau[1..100] de réel;
        i, j, max, N, ind : entier;
  Début
          Répéter
                  Ecrire(''saisir la dimension N du tableau'');
                  Lire(N);
          Jusqu'à (N>=0 ET N<=100);
          Si (N=0) alors
                  Ecrire(''le tableau est vide'');
          Sinon
                  // Saisie des éléments du tableau
                  Pour i ← 1 à N faire
                          Ecrire("entrer l'élément T1 ", i); Lire(T1[i]);
                  FinPour i
                  // Saisie des éléments du tableau avant le tri
                  Pour i ← 1 à N faire
                          Ecrire(T1[i]);
                  FinPour i
```

```
max ← 1000;
Pour i ← 1 à N faire
        ind ← 1;
        Pour j ← 2 à N faire
                Si(T1[ind]>T1[j]) alors
                        ind ← j;
                Finsi
        FinPour j
                T2[i] ← T1[ind];
                T1[ind] ← max;
 FinPour i
 // affichage  des éléments du tableau triés
Pour i ← 1 à N faire
          Ecrire(T2[i]);
FinPour i
Finsi
Fin
```