ALGORITHME
TABLEAUX

Mr KHATORY

Tableaux

Ensemble de données du même type

Exemple de problème :

Saisir une suite de nombres, puis afficher cette suite après avoir divisé tous les nombres par la valeur maximale de la suite.

⇒ Nécessité de conserver les nombres en mémoire

val | 132 | : Variable contenant une valeur variable

Val

132	0	8100	-641	841	8902	57	-21

: Variable contenant une collection de valeurs du même type

*Remarque : appeler cette variable **TabVal** plutôt que **Val***

2

Tableaux

Structure de données permettant d'effectuer un même traitement sur des données de même nature

tableau à **une** dimension 

tableau à **deux** dimensions 

Exemples d'applications:

- Ensemble de valeurs entières, réelles, booléennes,....
- Ensemble de noms (type chaîne)
- Ensemble de caractères (type caractère)
- Ensemble d'adresses (type Adresse : nom,adresse, num téléphone)
- Ensemble d'ouvrages

.....

3

Tableaux

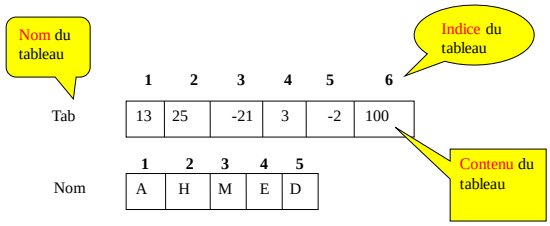
Traitements opérant sur des tableaux

On veut pouvoir :

- ✓ **créer** des tableaux
- ✓ **ranger** des valeurs dans un tableau
- ✓ **récupérer, consulter** des valeurs rangées dans un tableau
- ✓ **rechercher** si une valeur est dans un tableau
- ✓ **mettre à jour** des valeurs dans un tableau
- ✓ **modifier** la façon dont les valeurs sont rangées dans un tableau (par exemple : les trier de différentes manières)
- ✓ effectuer des **opérations entre tableaux** : comparaison de tableaux, multiplication,...
- ✓ ...

4

Tableaux



Remarques :

• Indices : en général, démarrage à 1, **mais en C++, démarrage à 0**

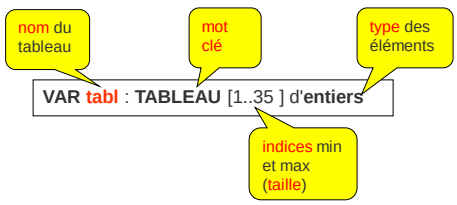
• Nombre d'octets occupés : dépend du type des valeurs enregistrées

5

Tableaux

Déclaration d'un tableau

Exemple : déclaration d'un tableau pouvant contenir jusqu'à 35 entiers



Autre exemple : déclaration d'un tableau qui contiendra les fréquences des températures comprises entre –40°C et 50°C

VAR Temperatures : **TABLEAU** [**1**..**91**] de **REEL**

Avec **Temperatures [1]** correspond à la fréquence de la température **-40°c**

Question: la fréquence de la température **25 °c** → **Temperatures [?]**

VAR Temperatures : **TABLEAU** [**-40**..**50**] de **REEL**
Temperatures [-30] → **Fréquence temp -30°C**

6

Tableaux

Définition d'un type tableau

TYPE <Nom> **=** <description>

Exemple : *déclaration d'un nouveau type Mot, tableau de 10 caractères*

TYPE Mot = TABLEAU [1..10] de **caractères**

VAR nom**,** verbe : **Mot**

Tableaux

Utilisation d'un tableau : par les indices

• Accès en lecture :

- x ← T[2] */*le contenu du tableau à l'indice 2 est placé dans x */*

- **Ecrire** (T[4]) */*le contenu du tableau à l'indice 4 est affiché à l' écran*/*

• Accès en écriture :

-T[3] ←18 */*la valeur 18 est placée dans le tableau à l'indice 3*/*

-**Lire** (T[5]) */*la valeur entrée par l'utilisateur est enregistrée dans le tableau à l'indice 5*/*

Tableaux

CONST TAILLE=50

TYPE TABL= **TABLEAU** [1..TAILLE] d'ENTIER

Ecrire la PROCEDURE **SAISIR(…)** qui permet de saisir un tableau T de **N** éléments (N ≤ TAILLE)

```
PROCEDURE SAISIR( VAR T  : TABL;N :ENTIER)
VAR i : ENTIER
DEBUT
        POUR i DE 1 A  N
                Lire (T[i])
        FINPOUR
FIN
```

Ecrire la PROCEDURE **AFFICHER(…)** qui permet d'afficher un tableau T de **N** éléments (N ≤ TAILLE)

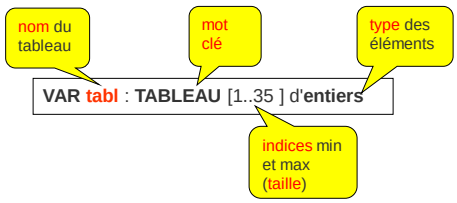
```
PROCEDURE AFFICHER (T :TABL; N :ENTIER)
VAR i :ENTIER
DEBUT
        POUR i DE 1 A  N
                ECRIRE (T[i])
        FINPOUR
FIN
```

9

Tableaux

Déclaration d'un tableau

Exemple : déclaration d'un tableau pouvant contenir jusqu'à 35 entiers



10

Tableaux

Définition d'un type tableau

TYPE <Nom> **=** <description>

Exemple : *déclaration d'un nouveau type Mot, tableau de 10 caractères*

TYPE Mot = TABLEAU [1..10] de **caractère**

VAR nom**,** verbe : **Mot**

Tableaux

CONST TAILLE=50

TYPE TABL= **TABLEAU** [1..TAILLE] d'entiers

Ecrire la PROCEDURE **SAISIR(…)** qui permet de saisir un tableau T de **N** éléments
(N ≤ TAILLE)

```
PROCEDURE SAISIR(VAR T  : TABL;N :entier)
VAR i : entier
DEBUT
        POUR i DE 1 A  N
                Lire (T[i])
        FINPOUR
FIN
```

Ecrire la PROCEDURE **AFFICHER(…)** qui permet d'afficher un tableau T de **N** éléments
(N ≤ TAILLE)

```
PROCEDURE AFFICHER (T :TABL; N :entier)
VAR i :entier
DEBUT
        POUR i DE 1 A  N
                ECRIRE (T[i])
        FINPOUR
FIN
```

TRI TABLEAUX

Trier des objets est à la fois un problème fondamental de l'algorithmique, comme étape de prétraitement d'algorithmes plus complexes ou pour ordonner des structures de données (à quoi servirait un annuaire non trié)

Nous présentons ici des tris simples possibles sur les tableaux:

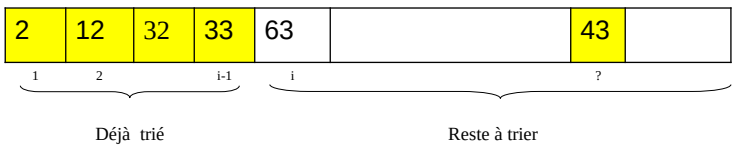
1. **Tri par sélection**
2. **Tri par Insertion**
3. **Tri à bulle**

TRI TABLEAUX

Tri par Sélection

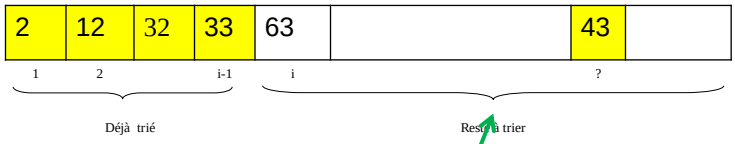
Algorithme de tri par sélection

L'idée du tri consiste à chaque étape à rechercher le plus **petit élément** non encore trié et à le placer à la suite des éléments déjà triés. A une étape i, les i − 1 plus petits éléments sont en place, et il nous faut sélectionner le ième élément à mettre en position i.



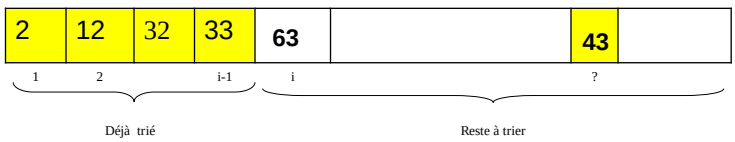
1. Chercher le plus petit élément du tableau restant
2. Echanger cet élément avec l'élément en position i

TRI TABLEAUX



ALGORITHME TriSélection
CONST TAILLE=50
TYPE TABL=TABLEAU[1..TAILLE] d'ENTIER
VAR T: TABL
N,i, pos :ENTIER
DEBUT
ECRIRE(" Entrer le nombre d'élément du tableau' < = ",TAILLE)
LIRE(N)
SAISIR(T,N)
POUR i DE 1 A N -1
FAIRE
/* Etapes de l'algorithme*/
pos ⇦ **TROUVERMIN(T ,i,N)** /* Plus petit élément du tableau restant*/
ECHANGER(T,i,pos) /* ECHANGER avec l'élément en position i*/
FINPOUR
AFFICHER(T,N)
FIN

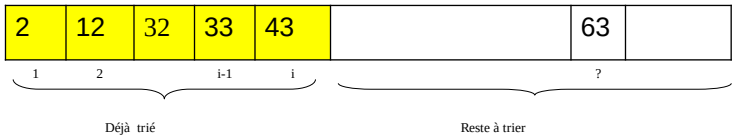
TRI TABLEAUX



FONCTION *TROUVERMIN*(T: TABL i,N: ENTIER):ENTIER
VAR i_Min :ENTIER
DEBUT
i_Min ⇦ i
POUR j DE i_Min +1 A N
FAIRE
SI T [j] < T[i_Min]
ALORS
i_Min ⇦ j
FINSI
FINPOUR
Retourner(i_Min)
FIN

PROCEDURE *ECHANGER*(Var T: TABL; i,j: ENTIER)
VAR Aux :ENTIER
DEBUT
Aux ⇦ T[i]
T[i] ⇦ T[j]
T[j] ⇦ Aux
FIN

TRI TABLEAUX



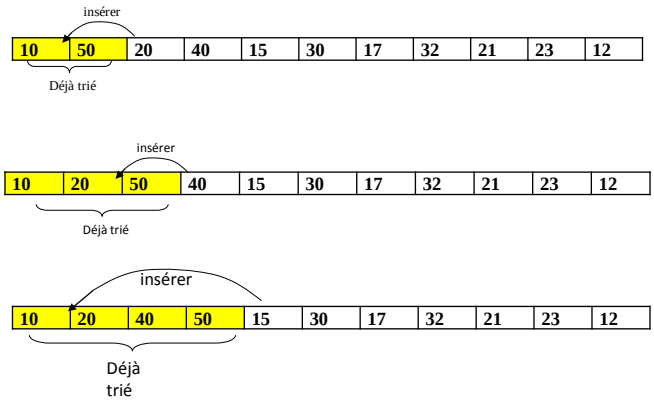

```
FONCTION TROUVERMIN( T: TABL i,N: ENTIER ):ENTIER
VAR i_Min :ENTIER
DEBUT
      i_Min ⇦ i
      POUR j DE i_Min +1 A N
      FAIRE
            SI T [j] < T[i_Min]
            ALORS
                  i_Min ⇦ j
            FINSI
      FINPOUR
      Retourner( i_Min)
FIN
```

```
PROCEDURE ECHANGER(VAR T: TABL; i,j: ENTIER
VAR Aux :ENTIER
DEBUT
      Aux ⇦ T[i]
      T[i] ⇦ T[j]
      T[j] ⇦ Aux
FIN
```

TRI TABLEAUX

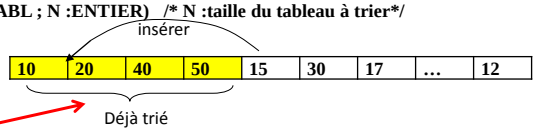
Écrire la Procedure **TRI_INSERTION** qui utilise la Procedure INSERER pour trier par ordre croissant les éléments d'un tableau de N éléments.

Méthode: Trier le tableau de gauche à droite en insérant à chaque fois l'élément i+1 dans le tableau (déjà trié) des i premiers éléments.



TRI TABLEAUX

```
PROCEDURE TRI_INSERTION( VAR T :TABL ; N :ENTIER)   /* N :taille du tableau à trier*/
VAR  i, indice              :ENTIER
Trouve                      :BOOLEEN
DEBUT
      POUR  i DE 2 A N
      FAIRE
            indice ⇦1
            Trouve ⇦ Faux
            TANTQUE (Non Trouve) ET  (indice <  i)
            FAIRE
                  SI T[indice] > T[i] ALORS
                        INSERER(T,indice,i)
                        Trouve ⇦ Vraie
                  FINSI
                  indice ⇦indice  +1
            FINTANTQUE
      FINPOUR
FIN
```




```
PROCEDURE  INSERER(var T :TABL ; indice,i :ENTIER)
VAR j :ENTIER
DEBUT
      Aux ⇦T[i]
      POUR j DE i-1 A indice Par pas -1
       FAIRE
            T[j+1] ⇦ T[j]
      FINPOUR
      T[indice] ⇦Aux
FIN
```

Décalage à droite des éléments

TRI TABLEAUX

```
PROCEDURE TRI_INSERTION( VAR T :TABL ; N :ENTIER)   /* N :taille du tableau à trier*/
VAR  i, indice              :ENTIER
Trouve                      :BOOLEEN
DEBUT
      POUR  i DE 2 A N
      FAIRE
            indice ⇦1
            Trouve ⇦ Faux
            TANTQUE (Non Trouve) ET  (indice <  i)
            FAIRE
                  SI T[indice] > T[i] ALORS
                        INSERER(T,indice,i)
                        Trouve ⇦ Vraie
                  FINSI
                  indice ⇦indice  + 1
            FINTANTQUE
      FINPOUR
FIN
```

```
POUR  i DE 2 A N
FAIRE
      indice ⇦1
      TANTQUE (T[indice] ≤T[i]  ET ( indice < i)
      FAIRE
            indice ⇦indice  +1
      FINTANTQUE
      SI (indice ≠i )
      ALORS INSERER(T,indice,i)
      FINSI
FINPOUR
```

Attention au cas de indice = i
Pas d'insertion !

TRI TABLEAUX

Tri à bulle

Le tri à bulle consiste à parcourir le tableau, par exemple de gauche à droite, en comparant les éléments côte à côte et en les permutant s' ils ne sont pas dans le bon ordre.

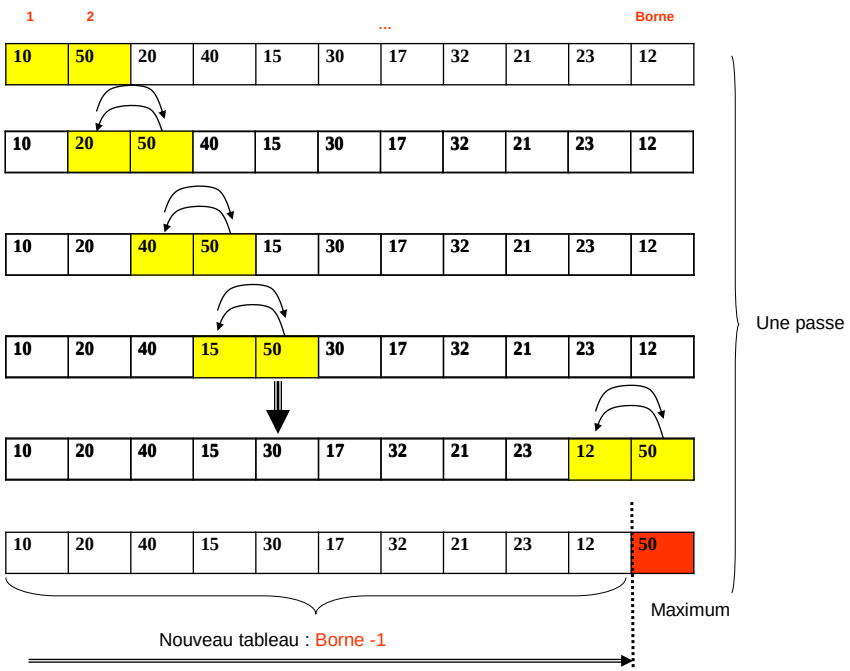
Au cours d'une passe du tableau, les plus grands éléments remontent de proche en proche vers la droite comme des bulles vers la surface.

On s'arrête dès que l'on détecte que le tableau est trié : si aucune permutation n'a été faite au cours d'une passe.

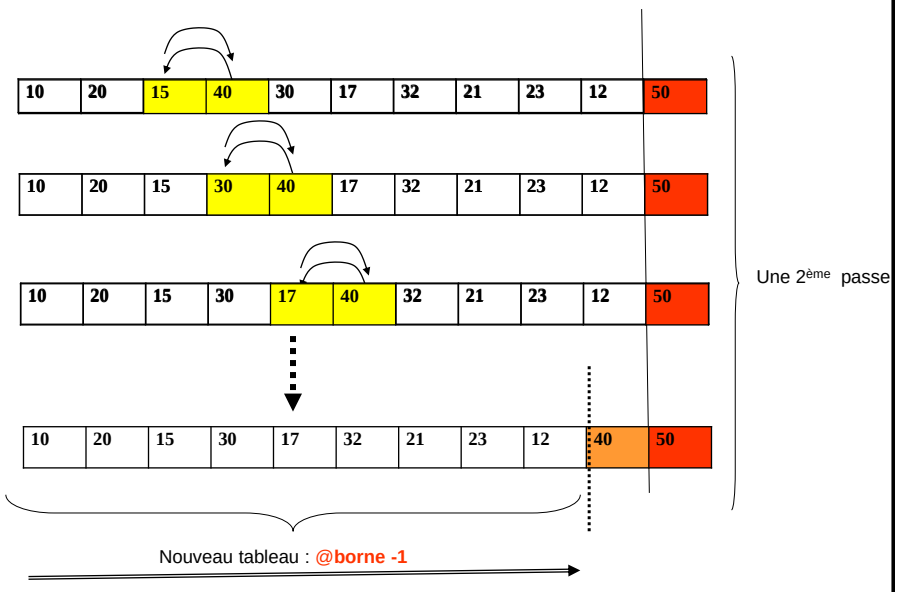
Tant que le tableau n'est pas trié
1. Effectuer une passe !

→

a. Parcourir le tableau
b. Si 2 éléments successifs ne sont pas dans le bon ordre, les échanger



Même méthode !! Pour le nouveau tableau de taille **borne -1**




```
ALGORITHME TriBulle
CONST   TAILLE=50
TYPE    TABL=TABLEAU [1..TAILLE] d'ENTIER
VAR     T          :  TABL
        i,N,borne  :ENTIER
        Est_trie   :BOOLEEN
DEBUT
    ECRIRE(" Entrer le nombre d'élément du tableau < = ",TAILLE)
    LIRE(N)
    SAISIR(T,N)
    Est_trie ⇦ faux                    /*Détecte si le tableau est trié*/
    borne ⇦  N
    TANTQUE Non(Est_trie )             /* Itération sur les passes*/
    FAIRE
        Est_Trie ⇦ vrai                /* Initialisation avant une passe*/
        POUR i  DE  1  A  borne-1      /* Effectue une passe*/
            Si T[i] > T[i+1]           /* Compare 2 éléments successifs*/
            Alors
                ECHANGER(T,i,i+1)
                Est_trie⇦faux
            FinSi
        FINPOUR
        borne⇦borne-1
    FINTANTQUE
    AFFICHER(T,N)
FIN
```