

# Algorithmique et Structures de données

L2 MPI-PC

**Dr Ousmane DIALLO**



# Chap6:

## Les tableaux et les chaînes de caractères

### 6.1. Tableaux à 1 dimension

#### 6.1.1 Définition

- ❑ Un **tableau** est une collection ordonnée de variables (appelées composantes du tableau ) ayant toutes le même type.
- ❑ Ces variables qui constituent le tableau sont stockées en mémoire centrale de manière contiguë (les unes à la suite des autres). On accède donc à chacune de ces variables individuellement à l'aide d'un indice. L'indice doit être de type scalaire et prendre ses valeurs dans un ensemble ordonné et fini.
- ❑ Un tableau **T** est donc défini par le **type des indices** et par le **type des composantes**. Les tableaux à *une dimension*, c'est-à-dire à un type d'indices, appelés encore en termes mathématiques **vecteurs**, sont composés d'un ensemble homogène d'éléments (pas des tableaux) de même type.

# Chap6:

## Les tableaux et les chaînes de caractères

### 6.1. 2. Déclaration d'un tableau

La déclaration d'un tableau s'effectue en donnant:

- son nom (identificateur de la variable tableau)
- le domaine de variation de l'indice délimité par une borne inférieure correspondant à l'indice minimal et la borne supérieure correspondant à l'indice maximal
- le type de ses composantes

**Syntaxe algo:**

**Variable** <Nom\_tab> :Tableau[<indice min>..<indice\_max>] de <type des composants>;

**Exemple**

```
variable tab_entier : Tableau[1..10] d'entiers;
```

**Syntaxe Pascal:**

**Var** <nom\_tab> : ARRAY [<indice min>..<indice max>] OF <type des composants> ;

**Exemple**

```
var tab_entier : ARRAY [1..10] OF integer;
```

# Chap6:

## Les tableaux et les chaînes de caractères

### NB:

La structure **ARRAY** n'est pas une structure "*dynamique*" mais "*statique*", c'est-à-dire une structure qui ne change pas de taille au cours de l'exécution du programme. Par conséquent, le nombre d'éléments d'un tableau doit être fixé à priori de manière définitive lors de sa définition.

### Exemple

```
CONST MaxElements = 100;  
TYPE Dim = 1..MaxElements;  
    Alphabet = 'A'..'Z';  
...  
VAR Boole: ARRAY [ Dim ] OF Boolean;  
    Frequency: ARRAY [ Alphabet ] OF Integer;
```

Il est préférable de représenter la structure d'un tableau moyennant la définition d'un type en écrivant:

```
TYPE <identificateur de type> = ARRAY[<type index>] OF  
<type composant>;
```

# Chap6:

## Les tableaux et les chaînes de caractères

Et ensuite en déclarera les variables de ce type tableau en écrivant :

**VAR** < liste variable >: < identificateur de type > ;

**Exemple**

```
CONST MaxElements = 100;
TYPE Dim = 1..MaxElements;
    Alphabet = 'A' .. 'Z';
    TBoole = ARRAY [ Dim ] OF Boolean;
    TFrequency = ARRAY [ Alphabet ] OF Integer;

.....
VAR Boole: TBoole;
    Frequency: TFrequency;
```

# Chap6:

## Les tableaux et les chaînes de caractères

### 6.1.3. Indexation

- ❑ Pour distinguer nommément les éléments du tableau, il faut tout simplement utiliser l'indexation, c'est-à-dire numérotter les éléments du tableau et ainsi leur accorder un **indice** ou un **index**. La valeur des indices doit être de type entier ou caractère.
- ❑ Le nom de la variable positionnée à l'indice  $i$  est donc:

**Nom\_du\_tableau [i];**

### Exemple:

- ❑ Une variable **T** de type tableau à une dimension peut être représentée comme suit:

1	2	3	4	5	6	7	8
10	0	1	4	7	2	6	25

Figure 6.1 – Tableau à une dimension (vecteur)

Et  $T[2]$  est la variable à la position 2, ...

# Chap6:

## Les tableaux et les chaînes de caractères

### 6.1.4. Initialisation des éléments d'un tableau

#### Initialisation (affectations)

```
Var T: array[1..5] of integer;  
Begin  
  T[1] := 1;  
  T[2] := 1;  
  T[3] := 1;  
  T[4] := 1;  
  T[5] := 1;  
End.
```

#### Initialisation (boucle)

```
Var T: array[1..5] of integer;  
  i : integer;  
Begin  
  for i:=1 to 5 do  
    T[i] := 1;  
  End.
```

#### Initialisation (saisie)

```
Var T: array[1..5] of integer;  
  i : integer;  
Begin  
  for i:=1 to 5 do  
    begin  
      writeln('Donner T[' ,i, ']:');  
      readln (T[i]);  
    end;  
  End.
```

# Chap6:

## Les tableaux et les chaînes de caractères

### 6.2. Tableaux à 2 dimensions

#### 6.2.1 Définition

- ❑ Un tableau à deux dimensions, c'est-à-dire à deux types d'indices, est un tableau dont le type des composantes est un type tableau de dimension un, donc, un vecteur. En termes mathématiques, un tableau à deux dimensions est appelé **une matrice**.
- ❑ La définition d'un tableau à deux dimensions peut s'effectuer de la manière suivante:

Syntaxe Pascale:

**TYPE** <identificateur> = **ARRAY**[1..M, 1..N] **OF** < type-composantes >;

Où < **type-composantes** > peut être un type quelconque, sauf le type tableau.

#### Exemple

```
CONST MaxLine = 10; {nombre maximal de lignes}
      MaxColumn = 20; {nombre maximal de colonnes}
TYPE TLine = 0..MaxLine;
      TColumn = 0..MaxColumn;
      Matrix = ARRAY [ TLine, TColumn ] OF Real;
...
VAR Mat: Matrix;
```



# Chap6:

## Les tableaux et les chaînes de caractères

- ❑ Si les variables **L** et **C** prennent respectivement leurs valeurs dans les intervalles **TLine** et **TColumn**, alors **Mat[L,C]** ou bien **Mat[L][C]** désigne la composante de la matrice située à la **L-ième** place dans la **C-ième** colonne, respectivement à la **C-ième** place dans la **L-ième** ligne.
- ❑ Donc, pour accéder à une composante d'une matrice, il est nécessaire de préciser deux indices, à savoir d'abord l'indice de ligne, puis l'indice de colonne.

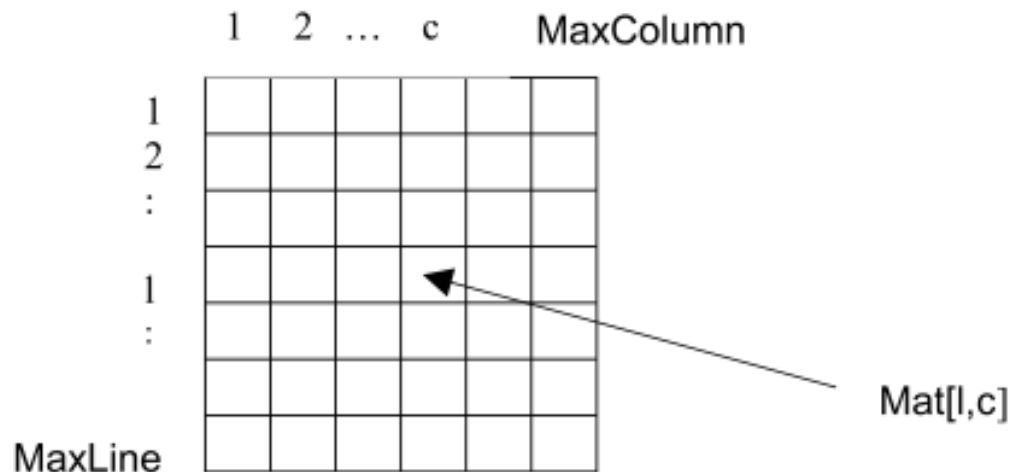


Figure 6.2 – Tableau à deux dimensions (matrice)

# Chap6:

## Les tableaux et les chaînes de caractères

### 6.3. Manipulations élémentaires de tableaux

#### 6.3.1 Création et affichage d'un tableau

- ❑ Créer un tableau équivalent à attribuer une valeur à chacune de ses composantes. La création peut se faire soit par une affectation, soit par une saisie des valeurs (voir section 6.1.4. , slide 7).
- ❑ L’Affichage à consiste parcourir le tableau à l’aide d’une boucle et à afficher ses différentes composantes

**Exemple:** Le programme suivant permet de "remplir" un tableau à l'aide d'une boucle Repeat-Until

```
Const
  Taille_max=10;
Type
  array[1..Taille_max] of Integer;
Var
  Tableau: TAB;
  indice: Integer;
Begin
  for indice:=1 to Taille_max do
    Tableau[indice]:=0;
  indice:=1;
  Repeat
    write('entrez le N°',indice,':');
    readln(Tableau[indice]);
    indice:=indice+1;
  until indice>Taille_max;
End.
```

# Chap6:

## Les tableaux et les chaînes de caractères

### 6.3.2 Maximum et minimum d'un tableau

- ❑ La recherche du maximum (minimum) consiste à mettre dans la variable max (min) le premier élément du tableau T, de parcourir ensuite le reste du tableau et de mettre à jour cette variable max (min) si l'élément en cours d'examen est supérieur (inférieur) au max (min) courant.

Soit les déclarations suivantes:

```
Const nmax=20;
```

```
Type tab = array[1..nmax]of integer;
```

```
Var t:tab; i, n, x, max : integer; trouve : boolean;
```

```
(*Recherche du max*)
```

```
max := t[1] ;
```

```
for i :=2 to nmax do
```

```
if t[i] > max then max := t[i];
```

```
writeln('Le max de t est :', max) ;
```

```
(*Recherche du min*)
```

```
min := t[1] ;
```

```
for i :=2 to nmax do
```

```
if t[i] < min then min := t[i];
```

```
writeln('Le min de t est :', min) ;
```

# Chap6:

## Les tableaux et les chaînes de caractères

### 6.3.3 Recherche séquentiel d'un élément d'un tableau

- ❑ La recherche séquentielle d'un élément **X** dans un vecteur **T** consiste à parcourir ce dernier et de trouver un indice **i** tel que **T[i]=a**.

(\*Recherche séquentielle d'un élément x\*)

```
i := 1;  
while (t[i]<>x) and (i<=n) do  
  i:=i+1;  
  if (i>n) then  
    write(x, 'n"appartient pas à t')  
  else  
    write(x, ' appartient à t');
```

(\*Recherche séquentielle d'un élément x avec variable booléenne\*)

```
i := 1; trouve:=false;  
while (i<=n) and (trouve=false) do  
  begin  
    trouve:=(t[i]=x);  
    i:=i+1;  
  end;  
if trouve then  
  write(x, ' appartient à t')  
else  
  write(x, ' n"appartient à t');
```

# Chap6:

## Les tableaux et les chaînes de caractères

### 6.3.4 Recherche dichotomique d'un élément dans un tableau ordonné

- ❑ La bonne idée est de comparer l'élément recherché avec l'élément situé au milieu du tableau:
  - si l'élément recherché est plus petit, on continue la recherche dans la première moitié du tableau;
  - s'il est plus grand, on continue dans la seconde moitié du tableau ;
  - On s'arrête si l'élément recherché a été trouvé ou si on ne peut plus diviser le tableau.
- ❑ Dans ce qui suit:
  - $x$  représente l'élément que l'on veut chercher
  - gauche, droite: délimitent la zone du tableau à l'intérieur de laquelle on veut rechercher  $x$

# Chap6:

## Les tableaux et les chaînes de caractères

(\*Recherche dichotomique d'un élément  $x$  dans un tableau ordonné\*)

.....

VARIABLES

gauche, droite, milieu : entier;

DEBUT

(\*saisie du tableau  $T$  de  $n$  éléments et de l'élément  $x$  à rechercher\*)

.....

gauche:=1; droite:=n; trouve :=faux

**TANT QUE** (gauche<=droite) **ET** non Trouve **FAIRE**

**DEBUT**

milieu:=(gauche +droite ) **DIV** 2;

**SI** ( $x = T[\text{milieu}]$ ) **ALORS**

trouve :=vrai

**SINON SI** ( $x < T[\text{milieu}]$ ) **ALORS**

droite:=milieu-1

**SINON**

gauche:=milieu+1;

**FIN;**

**SI** trouve **ALORS**

**Ecrire**( $x$ , ' appartient à T')

**SINON**

**Ecrire**( $x$ , ' n"appartient `a T');

**FIN.**

# Chap6:

## Les tableaux et les chaînes de caractères

### 6.3.5 Remplissage et affichage des éléments d'une matrice

(\*Remplir les éléments de la matrice **Mat** définie précédemment, cf diapo 8 \*)

**Begin**

```
for i:=0 to MaxLine do      {lignes}
  for j:=0 to MaxColumn do  {colonnes}
    readline (Mat[i,j]);
```

**End.**

(\*Affichage des éléments de la matrice **Mat** définie précédemment, cf diapo 8 \*)

**Begin**

```
for i:=0 to MaxLine do      {lignes}
  begin
    for j:=0 to MaxColumn do {colonnes}
      write (Mat[i,j]);
      writeln;
```

**end;**

**End.**

# Chap6:

## Les tableaux et les chaînes de caractères

### 6.3.6 Recherche d'un élément dans une matrice

(\*Recherche d'un élément **x** dans la matrice **Mat** définie précédemment, cf diapo 8 \*)

**Begin**

trouve:=false;

l:= 1;

**while** (l<=MaxLine) **and** (trouve=false) **do**

**begin**

        c:= 1;

**while** (c<=MaxColumn) **and** (trouve=false) **do**

**begin**

                trouve:=(Mat[l,c]=x);

                c:=c+1;

**end;**

        l:=l+1;

**end;**

**if** trouve **then**

**write**(x,' appartient à Mat')

**else**

**write**(x, ' n'appartient à Mat');

**End.**



# Chap6:

## Les tableaux et les chaînes de caractères

### 6.4. Chaînes de caractères

#### 6.4.1 Définition

- ❑ Une **chaîne de caractères** est une suite de caractères regroupés dans une même variable.
- ❑ En Pascal, une chaîne de caractères correspond à un **tableau de 255 caractères au maximum**, c.-à-d. une chaîne de 255 éléments de type **char**.

**Type string = array[1..255] of char**

- ❑ Le Type **string** est utilisé, mais ce n'est pas un type standard du pascal (certains compilateurs Pascal ne l'acceptent pas). Ce type permet de manipuler des chaînes de **longueur variable**.

#### Déclaration:

var S' : string;      {*crée une chaîne de caractères de 255 éléments de type **char***}

    S : string[12]; {*crée une chaîne de caractères de douze éléments de type **char***}

- ❑ Il est possible, comme dans tout tableau, d'accéder à un caractère particulier de la chaîne S, en écrivant simplement : **S[i]**.

#### Exemple

b	e	l	l	e		c	h	a	î	n	e
s[1]	s[2]	s[3]	s[4]	s[5]	s[6]	s[7]	s[8]	s[9]	s[10]	s[11]	s[12]

# Chap6:

## Les tableaux et les chaînes de caractères

- ❑ Il est possible de manipuler la chaîne de manière globale, sans passer élément par élément. Ceci est très utile pour des affectations ou des tests.

### Exemple

S := 'Bonjour';

S[4] := 's';

S[6] := 'i'

⇒ À présent, S vaut **'Bonsoir'**

S := 'ok';

⇒ À présent, S vaut **'ok'**

- On constate que la taille de S est variable (7 caractères au départ, 2 caractères ensuite)

### 6.4.2 Opérateurs et fonctions

- ❑ Il est possible de comparer des chaînes de caractères, on utilise alors les opérateurs:

=, <, >, <=, >=, <>

- ❑ Dans ce cas, l'ordre utilisé est l'ordre lexicographique (utilisation du code ASCII)

# Chap6:

## Les tableaux et les chaînes de caractères

### Exemples

```
(*Soit b un booléen ; b est-il vrai ou faux ? *)  
b := 'A la vanille' < 'Zut'; { vrai }  
b := 'bijou' < 'bidon'; { faux, c'est > car 'j' > 'd' }  
b := 'Bonjour' = 'bonjour'; { faux, c'est < car 'B' < 'b' }  
b := 'zim boum' > 'attends!'; { faux, c'est < car '' < 'a' }
```

```
(*Soit b un booléen ; b est-il vrai ou faux ? *)  
var  
S: String[4];  
begin  
S := '';  
Write(S) ; { rien n'est affiché: la chaîne est vide }  
S := 'toto' ;  
S[1] := 'm' ;  
Write(S) ; { la chaîne de caractère contient « moto » }  
end.
```

# Chap6:

## Les tableaux et les chaînes de caractères

### 6.4.2.1 Concaténation

$s := \text{concat}(s1, s2, s3...);$  (ou parfois  $s := s1 + s2 + s3...$ )

#### Exemple

```
s1 := 'bon';  
s2 := 'jour';  
s3 := s1 + s2 ;
```

*Nous obtenons alors s3 valant 'bonjour'*

### 6.4.2.2 Longueur

$\text{length}(\text{str}) \rightarrow \text{entier}$

#### Exemple

```
s1 := 'salut';  
s2 := 'bonjour';
```

*Nous obtenons alors  $\text{length}(s1)$  valant 5 et  $\text{length}(s2)$  valant 7*

# Chap6:

## Les tableaux et les chaînes de caractères

### Exemple

```
var
S: String;
begin
Readln (S);
Writeln(' ', S, ' ');
Writeln('longueur de la chaîne = ', length(S));
end.
```

### 6.4.2.3 Fonction POS

**pos(souschaîne, chaîne)**

$\Rightarrow$  position de la sous chaîne dans la chaîne.

### Exemple

```
var S: String;
begin
S := ' 123.5';
{ Convertit les espaces en zéros }
while Pos(' ', S) > 0 do
S[Pos(' ', S)] := '0';
end.
```

# Chap6:

## Les tableaux et les chaînes de caractères

### 6.4.2.4 Fonction COPY

**copy(source, index, compteur)**

⇒ string avec "compteur" caractères à partir de l'index.

#### Exemple

```
s:=copy('bonjour monsieur', 4, 4);
```

*Nous obtenons alors s valant 'jour'*

### 6.4.2.5 Procedure DELETE

**delete(chaine, debut, nb\_car)**

⇒ supprime le nombre de caractères spécifié par **nb\_car** à partir de la position indiquée par **debut**.

### 6.4.2.6 Procedure INSERT

**insert(chaine1, chaine2, position)**

⇒ insère **chaine1** dans **chaine2** à partir de la position indiquée par **position**.

#### Exemple

```
s:=insert('madame ', 'au revoir Fall', 11)
```

*Nous obtenons alors s valant 'au revoir madame Fall'*

# Chap6:

## Les tableaux et les chaînes de caractères

### 6.4.2.7 Fonction ORD

**ORD(caractère)**

$\Rightarrow$  entier (code ASCII).

**Exemple**

*ORD('A') vaut 65 et ORD('a') vaut 97*

### 6.4.2.8 Fonction CHR

**CHR(entier)**

$\Rightarrow$  caractère ayant ce code ASCII.

**Exemple**

*CHR(65) vaut 'A' et CHR(97) vaut 'a'*