Chap3 les variables indicées

Introduction

Les objets variable et constante sont des objets élémentaires, car ils ne peuvent contenir qu'une seule donnée à la fois.

Un objet structuré peut contenir un nombre fixe de données élémentaires, nous étudions dans ce chapitre les tableaux à une dimension (unidimensionnels) et les tableaux à deux dimensions (bidimensionnels).

Les tableaux permettent de manipuler plusieurs informations de même type, d'y accéder directement grâce à des indices : la 1_ere info, la 2_eme info, . . ., la i_eme info, . . .

On appelle souvent vecteur un tableau en une dimension et matrice un tableau à deux dimensions.

1. Les tableaux unidimensionnels (à une dimension)

1.1 ,Définition

Un tableau est une collection ordonnée d'éléments ayant tous le même type. On accède à chacun de ces éléments individuellement à l'aide d'un indice.

Représentation en mémoire :

Un tableau est un objet structuré formée de zones mémoire contiguës. Les données dans un tableau ont toutes le même type. Ce dernier est précisé à la déclaration avec la taille du tableau (le nombre d'éléments). Il peut être représenté sous la forme suivante :

	a	a + 1	a + 2	……	b – 1	b
t =	t[a]	t[a + 1]	t[a + 2]	…… ..	t[b - 1]	t[b]

a,a+1 ,a+2,. .....,b-1,b sont des indices pour indiquer le numéro de case.

t[a] ,t[a + 1],t[a + 2]..... t[b - 1], t[b] sont les éléments du tableau.

Accés à un élément d'un tableau :

En programmation un élément du tableau est désigné par le nom du tableau suivi de l'indice (le rang) de l 'élément dans le tableau, soit :

```
<Nom du Tableau>[<Indice de l'élément>]
```

Exemple :

Soit deux Tableaux *V1* et *V2* où *V1* contient *4* composantes et *V2* contient *6* composantes :

V1 :

1	*2*	*3*	*4*
12.50	*25.75*	*56.00*	*60.25*

V2 :

1	*2*	*3*	*4*	*5*	*6*
24.00	*38.25*	*28.00*	*70.25*	*63.00*	*96.25*

On a :

V1[1] = 12.55 ; V1[2] = 25.75 ; V1[3] = 56.00 ; V1[4] = 60.25

V2[1] = 24.00; V2[2] = 38.25; V2[3] = 28.00; V2[4] = 70.25; V2[5] = 63.00 ; V2[6] = 96.25 ;

Déclaration :

Un *tableau à une dimension* est une structure linéaire dans laquelle chaque élément est repéré par un seul indice. On l'appelle aussi un *vecteur*. D'où le mot clé ARRAY utilisé en PASCAL pour le déclarer.

Syntaxe de la déclaration d'un type tableau :

type identificateur_du_type_tableau = **array**[a..b] **of** type_éléments ;

où a est l'indice de la première case du tableau et b celui de la dernière, et où type est le type de chacune des valeurs qui seront contenues dans les cases du tableau. a et b sont forcément de type de type ordinal, c'est à dire qu'il doit prendre ses valeurs dans un ensemble fini et ordonné (l'indice ne peut donc pas être de type réel).

Déclaration des variables de ce type :

Var nom_variable_tableau : identificateur_du_type_tableau ;

exemples :

type tab1= **array[**2..9**] of** integer ; * tableau de 8 entiers dont les indices sont numérotés de 2 à 9 *

type tab2= **array[**1..10**] of** string ; *tableau de 10 variables de type chaines de caractères dont les indices sont numérotés de 1 à10*

Pour déclarer des variables de type tab:

Var t1,t2 : tab2 ;

Donc pour déclarer une variable de type tableau, il est préférable que le type correspondant ait été déclaré auparavant. Cependant on peut avoir une deuxième possibilité, sans déclarer le type comme suit :

Var t1,t2 : **array [**1..10**] of string;**

1.5.Opérations sur les tableaux unidimensionnels :

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,...

a. Ecriture dans un tableau (remplissage d'un tableau) et affichage des éléments d'un tableau:

On peut écrire de plusieurs façons dans un tableau :
- par affectation directe case par case : T[I] := valeur ;

Exemple 1
Le programme suivant permet de "remplir" un tableau à l'aide d'une boucle FOR

```
program remplir_tableau;
uses crt;
type TAB=array[1..10] of integer;
var  Tableau:TAB;
       indice: integer;

begin
  for indice := 1 to 10 do
  begin
       write('entrez l''élément N° ',indice,':');
       readln(Tableau[indice]);
   end ;
end.
```

Exemple2
Le programme suivant après avoir rempli un tableau permet d'afficher ses éléments à l'aide d'une boucle WHILE

```
program affichage_tableau;
uses crt;
var Tableau: array[1..10] of integer;
      indice: integer;
begin
  for indice := 1 to 10 do
  begin
       write('entrez l''élément N° ',indice,':');
       readln(Tableau[indice]);
   end ;
  indice := 1;
  while (indice <= 10) do
  begin
    writeln (Tableau[indice]);
    indice := indice + 1;
  end;
end.
```

b-Somme de deux vecteurs (tableaux) :

```
program Somme;
const  Taille_max=100;
type TAB=array[1..Taille_max] of integer;
var  v1,v2,v3:TAB;
      indice: integer;
begin
 for indice := 1 to taille_max do
 begin
       write('entrez l''élément N° ',indice,':');
       readln(v1[indice]);
  end;
 for indice := 1 to taille_max do
 begin
       write('entrez l''élément N° ',indice,':');
       readln(v2[indice]);
  end;
 for indice := 1 to taille_max do v3[indice] := v1[indice]+ v2[indice;

end.
```

c-Recherche d'un élément dans un vecteur :

```
program chercher ;
  type TAB=array[1..1000] of integer;
  var v1 : TAB;
        i : integer;
        Res : Boolean;
BEGIN
   Readln(x);
   Res:=false;
   i := 1;
   while (i <= 1000) and (not(res)) do
   begin
     res := (v[i]=x);
     i := i+1;
   end;
   if res then writeln ('l'élément appartient au tableau')
   else writeln ('le tableau ne contient pas ', x) ;
END;
```

On sort du while dans deux situations :
- Si i > vn, on a parcouru tout le tableau sans trouver x.
-Sinon i <= vn et v[i] = x : on a trouvé x.

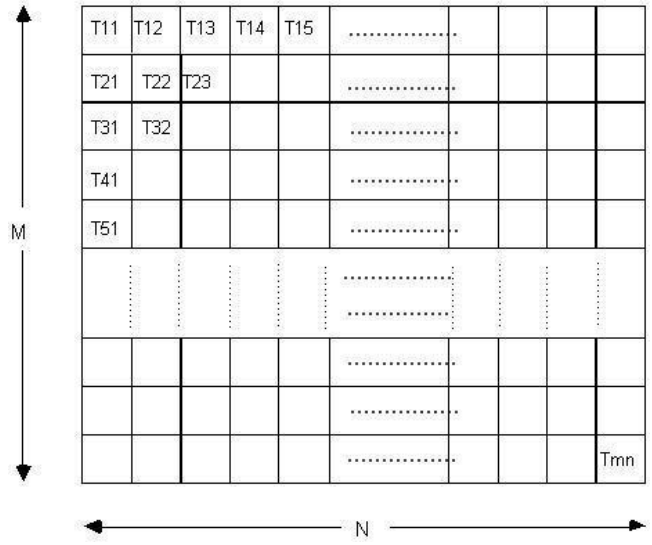
d-Tri d'un tableau :

```
Program TriSimple;
  uses wincrt;
  var
      T : Array[1..100] of real;
      I, J, N : integer;
      Z:real;
  Begin
       Clrscr ;
       N := 100 ;
       for i := 1 to N do read (T[i]);

      for i:=1 to (N-1) do
        for j:=(i+1) to N do
            if T[I] > T[J] then
            begin
                  Z:=T[I];
                  T[I]:=T[J];
                  T[J]:=Z;
            end;

      for i:=1 to N do write ( T[i],'   ');

  End.
```

2- Tableau à 2 dimensions :

Définition :

Un tableau de dimension 2 est parfois également appelé "MATRICE". Un tel tableau peut etre représenté sous la forme suivante :



Syntaxe : Declaration du type

type identificateur = array[1..M, 1..N] of type-elements;

L'accès a un élément d'un tableau T se fait par : T[i][j] ou T[i,j]

Exemples :

Type Tableau = array [1..10,1..20] of integer ;
Matrice = array [1..30,1..10] of real ;
Var A: matrice;
B: tableau;
Pour accéder à un élément de A il faut utiliser la notation : A[i,j] ; par exemple A[2,3] accède à l'élément de la ligne 2 et la colonne 3.

Opérations sur les matrices :

a-Lecture et affichage d'une matrice :

```
Program LectureAffichageMatrice;
  uses wincrt;
  var
    mat:array [1..10, 1..10] of real;
    N,M, i,j : integer;
Begin
        For i:=1 To 10  Do
              For j:=1 To 10 Do  Read(mat[i, j]);
        For i:=1 To 10  Do
        begin
              For j:=1 To 10   Do
              begin
                    Write(mat[i,j],'   ');
              end;
              writeln;
        end;
   End.
```

b- Initialisation d'une matrice unité :

Initialisation d'une matrice unité de dimension 10. Il s'agit donc d'une matrice à 10 colonnes et 10 lignes, ne comportant que des 0, sauf sur sa diagonale il n'y a que des 1.

```
program INITIALISATION_MATRICE ;
type Matrice = array [1..10,1..10] of integer ;
var i, j : integer;
     mat : Matrice;
begin
    for i := 1 to 10 do
    begin
      for j := 1 to 10 do
      begin
         If i = j then mat [i, j] := 1
        else mat [i, j] := 0 ;
        write (mat [i, j]);
     end ;
     writeln ;
 end ;
end.
```

produit d'une matrice par un vecteur :

```
Program ProduitMatVect;
  uses wincrt;
  var
    mat:array [1..10, 1..10] of real;
    vect, p:array[1..10] of real;
       i,j : integer;
Begin
      Clrscr ;
      For i:=1 To 10 Do
            For j:=1 To 10 Do Read(mat[i, j]);
      For i:=1 To 10 Do Read(vect[i]);
      For i:=1 To 10 Do
      begin
        p[i]=0;
        For j:=1 To 10 Do p[i]:= p[i]+ mat[i,j]*vect[j];

      end;
      For i:=1 To 10 Do
        write(p[i]);
End.
```

Compter le nombres d'éléments négatifs,positifs et nuls dans une matrice :

```
Program compteurPNN;
  uses wincrt;
  var
    mat:array [1..10, 1..10] of real;
    i,j : integer;
    nbP, nbN, nbNul: integer;
Begin
     Clrscr ;
      For i:=1 To 10 Do
            For j:=1 To 10 Do Read(mat[i, j]);
      nbP:=0; nbN:=0;
      nbNul:=0;
      For i:=1 To 10 Do
            For j:=1 To 10 Do
              if mat[i,j]>0 then nbP:=nbP+1
              else
                if mat[i,j]<0 then nbN:=nbN+1
                else nbNul:=nbNul+1;

      Write(nbP, nbN, nbNul);
End.
```

3-Exercices sur les tableaux

1. Ecrire un algorithme qui retourne l'indice de l'élément le plus petit contenu dans un tableau.
2. Ecrire un algorithme qui retourne l'indice de l'élément le plus grand contenu dans un tableau.
3. Ecrire un algorithme qui calcule la somme des valeurs contenues dans un tableau.
4. Ecrire un algorithme qui calcule la moyenne des valeurs contenues dans un tableau.
5. Ecrire un algorithme qui indique si un élément donné appartient à un tableau ou pas
6. Ecrire un algorithme qui indique si oui ou non un élément donné appartient à un tableau et si oui à quelle place (indice) il se trouve dans le tableau.
7. Ecrire un algorithme qui détermine l'image d'un tableau après rotation vers la droite(le 1er élément va en 2ème position, le 2ème passe en 3ème position ...etc, le dernier passe en 1ère position).
8. Ecrire un algorithme qui détermine l'image d'un tableau après rotation vers la gauche (le 1er va en dernière position, le 2ème va en 1ère position, le 3ème va en $2^{ème}$ position ... etc)
9. Ecrire un algorithme qui affiche une matrice (tableau à 2 dimensions) quelconque ligne par ligne.
10. Ecrire un algorithme qui effectue la somme de 2 matrices représentées par des tableaux à 2 dimensions.