4.Tableaux

4.1. Quelques notions: (rappel)

4.1.1. Identificateur :

- Un identificateur désigne le nom d'une variable, constante, type de données, procédure ou fonction…

4.1.2. Variable :

- Une variable possède un nom, un type, et une valeur.

4.1.3. Type :

- Les données peuvent êtres des types simples ou structurés, en plus il ya la possibilité de définir de nouveau types de données.
  - ✓ Types simples :
    Exp : entier, réel, caractère, booléen
  - ✓ Types structurés :
    Exp : tableaux, chaine de caractères, enregistrement…

4.2. Tableaux

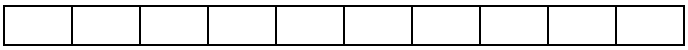
- Un tableau est une structure de données regroupant un nombre fixe de variables de même type.
- Un tableau peut être à une dimension (vecteur), à deux dimensions (matrice), et a plusieurs dimensions (tableau multidimensionnel).

4.2.1. Vecteurs : (tableau à une dimension)

Déclaration

- En algorithmique, on déclare un vecteur comme suit:
  Nom-Vect [taille] : tableau de **type-elements** ;

  Exp: V [20] : tableau de entiers ;



- On peut déclarer un vecteur comme suit :
  CONST n←10;
  V [n] : tableau de entiers ;

Représentation d'un Vecteur :

12.5	3.9	0.8	1.13	2.0	0.0	5.0	1.2	0.1	0.5
I=1	I=2	I=3	I= 4	I=5	I=6	I=7	I=8	I=9	I=10

V [5] = 2.0
L'indice peut être :

- Une Valeur : V [5]
- Une variable : V [i]
- Une expression : V [i*2]

Ecriture dans un vecteur (modification):

- Il existe deux méthodes pour écrire ou remplir la valeur d'une case d'un vecteur :
  1) par instruction d'affectation: **Exp :** V[2] ← 15; V[2*i] ← X+15;
  2) par instruction de lecture: **Exp** : Lire (V[2]);

- Si la taille du vecteur est devine grande, on utilise les boucles.

Exp :

```
Lire (V[1]) ;
Lire (V[2]) ;                          Pour i allant de 1 à 10 Faire
Lire (V[3]) ;          >>>>>>>             Lire (V [i]) ;
…….                                     Fin Pour
Lire (V[10]) ;
```

Lecture dans un vecteur :

- Pour lire une valeur d'une case d'un vecteur, on procède comme suit :

1) Utilisation dans une expression : **Exp :** X← (Notes[1] + Notes[2]) / 2 ;

2) Comparaison :

```
Exp : Si ( Notes[ 1 ] >= 10) alors ;
          Ecrire("Admis") ;
      Sinon
          Ecrire("Ajourné") ;
      FinSi
```

3) Par instruction d'écriture (affichage) : **Exp :** écrire (V[2]) ;

- Si la taille du vecteur devient grande, on utilise les boucles.

Exemple :

Écrire un algorithme qui permet de lire les moyennes de 25 étudiants, puis calcule la différence entre la moyenne de chaque étudiant avec celle de la moyenne de groupe ?

```
ALGORITHME  Exp_Vect
            VMOY [25] : tableau de réel;
            i : entier ;
            SMOY, MOYG : réel ;
Début
  // Remplir (lire) le tableau (vecteur)



  // Calculer la moyenne de groupe
  SMOY ← 0 ;
  Pour i allant de 1 à n Faire
       SMOY ← SMOY+TMOY[i] ;
  Fin Faire
  MOYG ← SMOY / N ;
  Ecrire (" la moyenne du groupe est ", MOYG) ;
  //Calcul de la différence entre la moyenne de groupe et celle de l'étudiant
  Pour i allant de 1 à n Faire
     Ecrire (" la différence de la moyenne du groupe et celle de l'étudiant ", i," est= ", MOYG - VMOY[i]) ;
  Fin pour
Fin.
```

On peut écrire les deux premières boucles en une seule. Simplifier alors cet algorithme.

Remarque :

- La taille d'un tableau est fixe et ne peut être donc changée dans un programme : il en résulte deux défauts :
  - ✓ Si on limite trop la taille d'un tableau on risque le dépassement de capacité.
  - ✓ La place mémoire réservée est insuffisante pour recevoir toutes les données

4.3. Les méthodes de recherche dans un vecteur :

4.3.1. Recherche du maximum d'un vecteur :

```
Algorithme Rech_max
   Const tailleM ←100 ;
   Vect [tailleM] : tableaux de réel ;
   Max : réel ;
Début
   // On suppose que les éléments du vecteur ont déjà été lus.
    Max ← vect[1];
    Pour i allant de 2 à n faire
        Si vect[i] > max alors
              Max ← vect[i] ;
        Finsi
    Finpour;
    Ecrire ('le maximum est ', max) ;
Fin.
```

4.3.2. Recherche séquentielle :

- L'une des premières opérations sur les tableaux est la recherche d'un élément, son nombre d'apparition, sa ou bien ses positions.
- Pour cela, on doit parcourir tout le vecteur élément par élément et le comparer avec la valeur de l'élément à chercher.

Applications :

1. Chercher la position de la première occurrence d'un élément 5 dans un vecteur V contenant n éléments entiers ?

```
Algorithme recherche1
    Const  n←10 ;
    V[n] : Tableau de entier ;
     i : entier ;
Début
        // On suppose que les éléments du vecteur ont déjà été lus.
      // Chercher la position de la première occurrence de l'élément 5
      i ←1 ;
      Tant que (i<= n et V[i] ≠5) faire
          i ←i+1 ;
      Fin tant que
      Si (i>n) alors
```

```
            Ecrire ("Elément introuvable") ;
        Sinon
            Ecrie ("La position de l'élément est :", i) ;
        Finsi
Fin.
```

2. Chercher le nombre d'apparition de l'élément 5 dans un vecteur V contenant n éléments, ainsi que les positions des occurrences de cet élément ?

```
Algorithme recherche2
    Const  n←10 ;
    V[n] : Tableau de entier ;
     i, nba : entier ;
Début
    // Remplir le tableau (vecteur)
     Pour i de 1 à n Faire
        Ecrire (" donner l'élément N° ", i) ;
        Lire (V [i]) ;
     Fin pour
     i ←1 ;
     compt←0 ;
    Tant que (i<=n) faire
            Si (V[i]=5) alors
                compt ←compt+1 ;
                Ecrire (" la position d'occurrence 5 est ", i) ;
             finsi
            i ←i+1 ;
     Fin tant que
     Ecrire ("le nombre d'occurrence de 5 est :", compt) ;
Fin.
```

4.3.3. Recherche dichotomique :

- Ce type de recherche s'effectue dans un tableau ordonné :

1) On divise le tableau en deux parties sensiblement égales,
2) On compare la valeur à chercher avec l'élément du milieu,
3) Si elles ne sont pas égales, on s'intéresse uniquement à la partie contenant les éléments voulus et on délaisse l'autre partie.
4) On recommence ces 3 étapes jusqu'à avoir un seul élément à comparer.

Application :

On suppose qu'on dispose d'un vecteur **V** de **n** éléments. On veut chercher la valeur **Val** ?

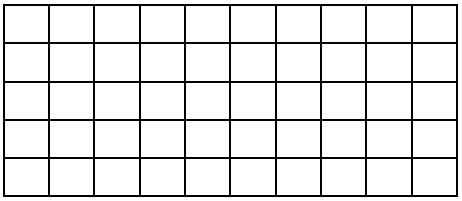
```
Algorithme  rech_dich
      Const  n←100 ;
      V[n] : Tableau de entier ;
      Iinf, Isup, Imil , Val : entier ;
      Trouv : Booléen;
Début
     Iinf ← 1 ;   Isup ← n ;
    Trouv  ← faux ;
    Tant que (Iinf <= Isup) et (Trouv = faux) Faire
          Imil ← (Iinf+Isup) div 2 ;
          Si (V[Imil] = Val) Alors
               Trouv ← vrai;
          Sinon
               Si (V [Imil] < Val) Alors
                    Iinf ← Imil + 1 ;
               Sinon
                    Isup ← Imil -1 ;
               Fin Si
          Fin Si
      Fin tant que
      Si (Trouv = vrai) Alors
           Ecrire (Val, "existe à la position" , Imil) ;
      Sinon
           Ecrire (Val, "n'existe pas dans V'') ;
      Fin Si
Fin.
```

Remarque : d'autres applications ont été vues et traitées en cours.

4.4. Matrices : (tableau à deux dimensions)

Déclaration :

- En algorithmique, on déclare une matrice comme suit :
  nom_matrice [**nbr_lignes**, **nbr_colonnes**] : **tableau** de type_elements
  Exp : M [5, 10] : tableau de réel ;



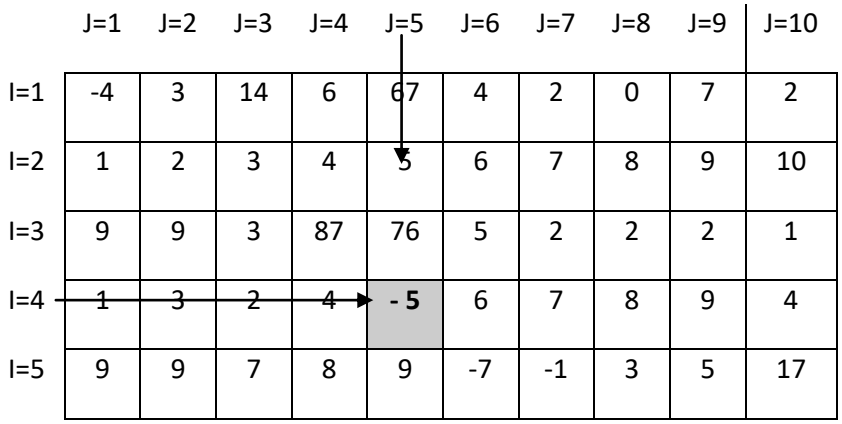
- On peut déclarer une matrice comme suit :

```
CONST  n←5, m←10;
M [n, m] : tableau de entiers ;
```

1)

```
n, m : entier ;
Mat [n, m] : tableau de réel ;
```

Représentation d'une Matrice:



	J=1	J=2	J=3	J=4	J=5	J=6	J=7	J=8	J=9	J=10
I=1	-4	3	14	6	67	4	2	0	7	2
I=2	1	2	3	4	5	6	7	8	9	10
I=3	9	9	3	87	76	5	2	2	2	1
I=4	1	3	2	4	**- 5**	6	7	8	9	4
I=5	9	9	7	8	9	-7	-1	3	5	17

- Chaque élément est identifié par deux indices : l'indice de la ligne i et l'indice de la colonne j ;
- L'élément d'indice [i,j] est celui du croisement de la ligne i avec la colonne j
- Par exemple M [4,5] est -5.

L'écriture dans la matrice (La modification) :

1) par instruction de lecture : Lire(Mat[8, 2]) (saisie de la valeur par clavier);
2) par instruction d'affectation : Mat[7, 2] ← 11.75 ;

La lecture d'un élément de la matrice (La consultation) :

1) Utilisation dans une expression : X← (Mat[5, 1] + Mat[5, 2]) / 2 ;
2) Comparaison : Si (**Note[6, 2] >= 10**) alors /

```
                Ecrire("Module acquis") ;
        Sinon
                Ecrire("Module n'est pas acquis") ;
         FinSi
```

Exemple :

Soit Mat (n, m) une matrice de nxm éléments réels. Ecrire un algorithme qui permet de calculer le plus grand (max) et le plus petit (min) élément de la matrice ?

```
Algorithme maxmin
    Const n←10, m←12 ;
     Mat [n, m] : tableau de réel;
     max, min : réel ;
     i, j : entier ;
Début
          //Lecture des éléments de la matrice
    Pour i←1 à n faire
       Pour j←1 à m faire
           Lire (mat [i, j] );
       Fin Pour
    Fin Pour
        // calcule de plus grand (max) et le plus petit (min)
     max← mat [1, 1] ; min← mat [1,1] ;
     Pour i←1 à n faire
       Pour j←1 à m faire
          Si (mat [i, j] >max) alors    max ← mat [i, j] ;   FinSi
          Si (mat [i, j] <min) alors    min ← mat [i, j] ;   FinSi
        Fin Pour
```

```
    Fin Pour
      Ecrire ("la plus grande valeur de la matrice", max) ;
      Ecrire ("la plus petite valeur de la matrice", min) ;
Fin.
```

Remarque :

- Une ***matrice carrée*** est une matrice dont le nombre de lignes est égal au nombre de colonnes.
- Une telle matrice a une ***diagonale principale*** (tous les éléments pour lesquels i=j).
- Les éléments supérieurs à la diagonale ont leurs indices **i<j** et Ceux inférieurs à la diagonale ont leurs indices **i>j**.

4.5. Les tableaux multidimensionnels : (Les tableaux à plusieurs dimensions)

La manipulation des tableaux à plusieurs dimensions est similaire à celle des matrices, on ajoute seulement d'autres dimensions.

Déclaration :

nom_tableau [**nbr_elm_d1**, **nbr_ elm_d2, nbr_ elm_d3,……..**] : **tableau** de type_elements ;

Exemple :

Déclaration d'un tableau a 3 dimensions :

M[10,20,30] : tableau d'entier ;