Les structures de données statiques:

Les tableaux et les matrices

Les tableaux

1) Introduction:

Exemple :

1- Ecrire un algorithme permettant de calculer la moyenne générale d'une classe de 30 étudiants.

```
Debut
    ……
    Smoy:=0;
    Pour i ← de 1 à 30 faire
        //Calculer la moyen  « moy » pour un seul étudiant
        Smoy := Smoy+moy;
    FinPour;
    moyG:= Smoy/30;
Fin
```

2- Effectuer leur classement.

Les tableaux

- **Conclusion :**

On ne peut pas effectuer le classement, parce qu'on ne garde pas les moyennes précédentes et la variable 'moy' contient seulement la dernière valeur de moyenne.

- **Solution :**

définir un ***objet structuré*** (tableau) qu'on appelle 'moy' qui sera divisé en 30 parties équitables, indicées de 1 à 30. Où, chaque partie va contenir la moyenne d'un étudiant.

2) Définition :

Un tableau à une dimension est un objet structuré **formé de cellules contiguës** accessibles directement à l'aide d'un « indice » qui précise la position (rang) d'un élément dans le tableau.

Un tableau peut être vu comme une liste d'éléments obligatoirement de même type. On le représente souvent comme une suite de cases contenant chacune une valeur.

Caractéristiques d'un tableau

Un tableau :

1. Possède un **nom**
2. Possède un **nombre d'éléments** ou de cases qui représente sa taille.
3. Est une **variable complexe** car il est lui-même constitué de variables.
4. Occupe une **zone mémoire contigüe**.
5. Possède une **taille finie et fixe**.
6. Tous ses éléments sont du **même type**.

MOY

contenu →	15	11,5	12,75	07,5	……	05	13,5
indice →	1	2	3	4	……	N-1	N

Les tableaux

Déclaration :

La déclaration d'un tableau est identique à celle d'une variable ordinaire, sauf qu'elle doit être suivie de sa taille, c'est-à-dire de son nombre d'éléments.

Syntaxe :

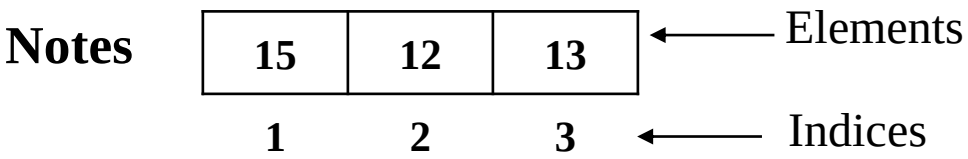
Var

Vect : **tableau** [Intervalle de l'indice] de <type élémentaire>;

Exemple :

Var

Notes **: Tableau [**1..3 **]** d'entiers;



Les tableaux

3) Manipulation des éléments d'un tableau :

3.1) L'accès à un tableau : (en lecture ou en écriture)

1. Pour désigner un élément du tableau, on indique le nom du tableau suivie de l'indice de l'élément entre crochets ex: **Notes[3]** .

2. La sélection d'un élément du tableau est décrire par une **notation indicée** : si "i" est un entier de l'intervalle de définition [1..N] d'un tableau T, T[i] désigne la valeur de T associée à "i". On dit que i est un ***indice*** de T.

Exemple:

Notes[3] ← 5 : affecte la valeur 5 à la 3ième case du tableau).

Lire(Notes[1]) : met **l'entier** saisi par l'utilisateur dans la **première case** du tableau.

Ecrire (Notes[2]) : affiche la valeur du 2iè case du tableau

Les tableaux

- Si *T1* et *T2* sont deux tableaux de **même taille** et **du même type**, on peut écrire l'instruction suivante : **T1 ← T2**.

- **Pour afficher les valeurs** de chaque case **de tableau**. Le mieux est de se servir **des boucles**. La boucle parcourt le tableau à l'aide d'une variable appelée "compteur" .

- La lecture (ou l'écriture ne peut se faire qu'élément par élément en parcourant tout le tableau (c'est pour cette raison qu'on utilise les boucles)

- **Exemple :** écrire un algorithme permettant d'effectuer la lecture et l'affichage d'un tableau 'Tab' de 20 cases entiers (avec la boucle pour).

Les tableaux

```
Algorithme affichage;
Var
        Tab = Tableau [1..20 ] d'entier;
        i : entier;
Début
   Ecrire (" Entrez les éléments du tableau") ;
   Pour i ← 1 à 20 faire
        lire (Tab[i]) ;
   FinPour;
   Pour i ← de 1 à 20 faire
        écrire (Tab[i]) ;
   FinPour;
Fin
```

Les tableaux

3.2) La recherche dans un tableau : On a souvent besoin de rechercher, dans un grand tableau, la position d'un élément donné. Un point particulier à ne pas oublier pour tous les algorithmes est le traitement du cas où l'élément cherché n'est pas dans le tableau.

a) Recherche séquentielle : Il suffit de parcourir le tableau élément par élément du début vers la fin, et de comparer chaque élément de tableau avec l'élément à chercher.

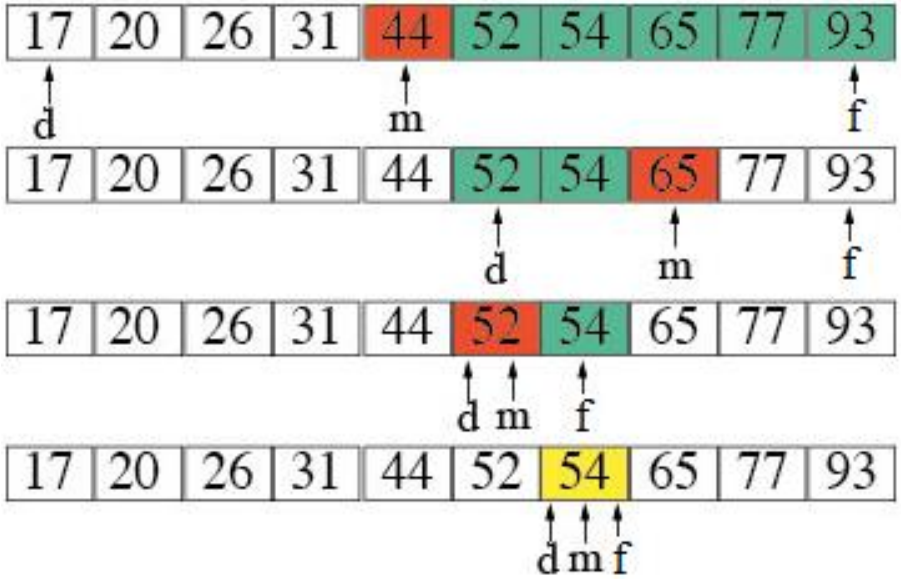
- **Exemple :** écrire un algorithme permettant de chercher la position de la première occurrence d'un élément "**elt**" dans un tableau '**T**' contenant '**N**' éléments. (on suppose que le tableau est définit).

Les tableaux

```
Algorithme Recherche_Séq ;
Const  N= 100 ;
Var  T : Tableau [1..N] de caractère ;
       i : entier ;
       elt : caractère ;
       trouve : booléen ;
Début
      écrire (‘ donner l’élément à chercher’) ;
      Lire (elt);
      trouve ← faux ;
      i ← 1;
      Tant que (trouve = faux) et (i<=N) Faire
            Si  T[i] = ‘elt’  Alors      trouve ← vrai
            Sinon      i ← i+1;
            FinSi ;
      FinTant que;
      Si  trouve = vrai  Alors     Ecrire (i) ;
      Sinon            Ecrire (‘ l’élément recherché n’existe pas dans le
      tableau’) ;
      FinSi;
Fin.
```

Les tableaux

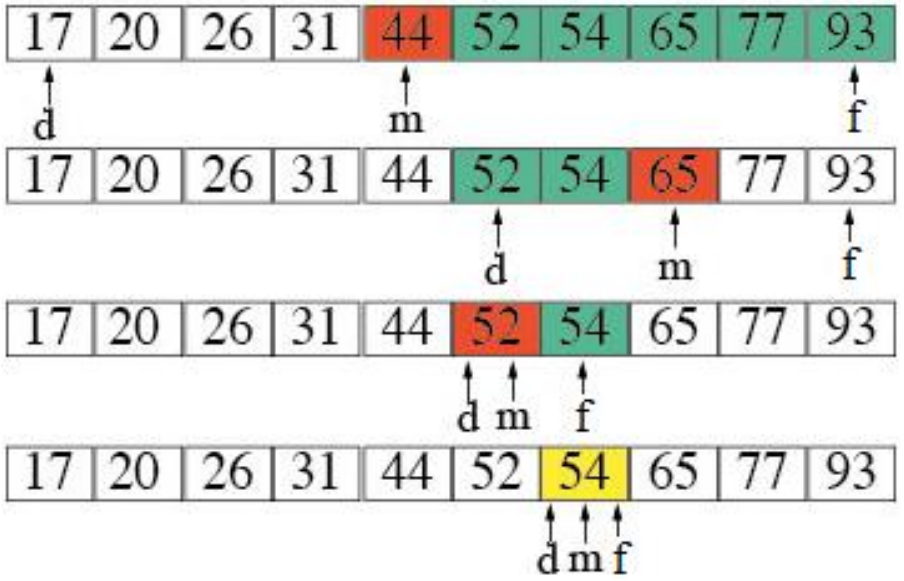
b) Recherche dichotomique : Dans le cas d'un tableau trié, on peut réduire le nombre de lectures, en cherchant à limiter l'espace de recherche. On compare la valeur cherchée à l'élément central du tableau, si ce n'est pas la bonne, un test permet de trouver dans quelle moitié du tableau on trouvera la valeur. On continue récursivement jusqu'à un sous-tableau de taille 1.



Les tableaux

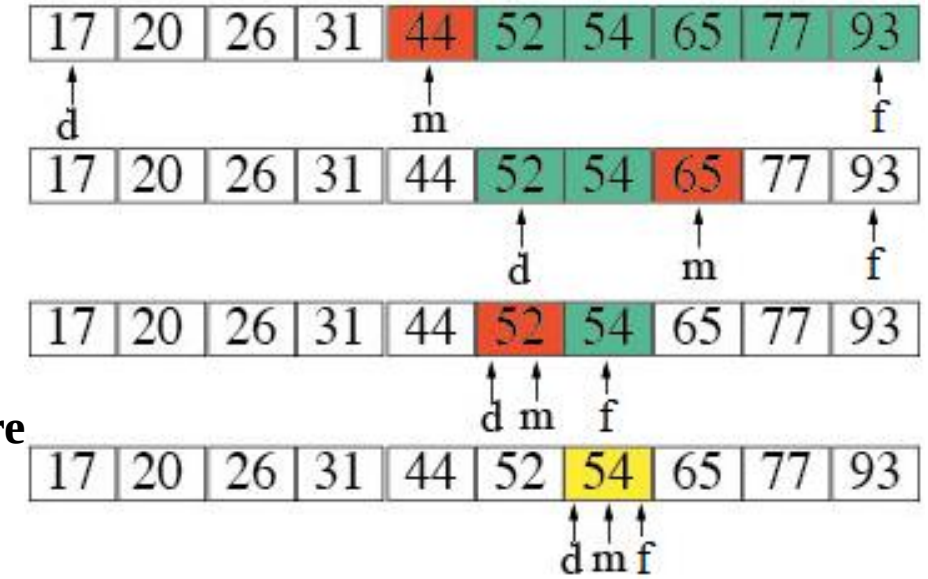
- **Principe :**

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 sont différents, on s'intéresse uniquement à la partie contenant l'élément voulu.
4. On répète ces 3 étapes jusqu'à avoir un seul élément à comparer.



Les tableaux

```
Algorithme Recherche_Bin ;
Const N= 100 ;
Var  T : Tableau [1..N] de entier ;
                f, m , d , i : entier ;
                trouve : booléen ;
Début
   Pour i ← 1 à N Faire
      Lire (T[i]) ;
   FinPour ;
   trouve ← faux ;   d ← 1 ;   f  ← N ;
   Tant que (d<= f) et (non trouve) Faire
       m ← (d + f) Div 2 ;
       Si (T[m] > 54 ) Alors    f ← m – 1
       Sinon  Si T[m] < 54 Alors   d ← m + 1
                   Sinon    trouve  ← vrai ;
                   FinSi ;
       FinSi ;
   Fin_Tant que ;
   Si          trouve = vrai  Alors  Ecrire (m);
   Sinon     Ecrire (‘ l’élément recherché n’existe pas dans le tableau’) ;
   FinSi;
Fin.
```



Les tableaux

3.3) Le tri d'un tableau : trié un tableau c'est le fait de ranger les éléments d'un tableau en ordre croissant ou décroissant.

Dans tous ce qui suit, on suppose que l'on trie des nombres entiers et que ceux-ci se trouvent dans un tableau T.

- **Tri par sélection:**

  Le principe consiste à fixer le premier élément du vecteur comme minimum, puis le comparer aux autres éléments du tableau, s'il existe un élément inférieur au minimum fixé, alors on fait la permutation, l'élément en question prend la place du minimum et on recommence jusqu'à arriver au dernier élément du tableau.

Les tableaux

Principe:

1. Parcourir le tableau *T* de la gauche vers la droite et rechercher l'élément le plus petit, qui est alors permuté avec *T[1]*.
2. Effectuer le premier traitement avec l'élément suivant (c.à.d. commencer la recherche à partir de *T[2]*). Répéter le même traitement en excluant à chaque itération l'élément trié, jusqu'à ce que tous les éléments soient classés.

❑ Il existe d'autres méthodes de tri parmi elles on peut citer:

- Tri par échange (à bulles).
- Tri par insertion.

Les tableaux

```
Algorithme Tri_selection ;
Const N=100 ;
Var  T : Tableau [1..N] de entier ;
        i, k , X : entier ;
Début
  Pour i ← 1 à N Faire
     Lire (T[i]) ;
  FinPour ;
  Pour i ← 1 à N-1 Faire
   Pour k ← i+1 à N Faire
     Si (T[i] > T[k] ) alors
        X ← T[k] ;
        T[k] ← T[i] ;
        T[i] ← X ;
     FinSi ;
   Fin_Pour ;
  Fin_Pour ;
  Pour i ← 1 à N Faire
     Ecrire (T[i]) ;
  FinPour ;
Fin.
```

L = [3 , 1 , 7 , 9 , 4 , 12 , 5]

Etape 0 ➔ L = [**1** , **3** , 7 , 9 , 4 , 12 , 5]

Etape 1 ➔ L = [1 , **3** , 7 , 9 , 4 , 12 , 5]

Etape 2 ➔ L = [1 , 3 , **4** , 9 , **7** , 12 , 5]

Etape 3 ➔ L = [1 , 3 , 4 , **5** , 7 , 12 , **9**]

Etape 4 ➔ L = [1 , 3 , 4 , 5 , **7** , 12 , 9]

Etape 5 ➔ L = [1 , 3 , 4 , 5 , 7 , **9** , **12**]

Les tableaux

NB: Les opérations de bases sur les tableaux ***de même type*** sont :

- **L'affectation (←)** qui **copie tous les éléments du tableau** (opérande droite) **dans un autre** (opérande gauche)
- **L'égalité (=)** qui permet de savoir si deux tableaux de même type possèdent des éléments de même valeur ($\forall i, a[i]=b[i]$).
- **L'inégalité (≠)** qui permet de savoir si deux tableaux de même type possèdent au moins un élément différent ($\exists i, a[i] \neq b[i]$)

Les tableaux à deux dimensions

1) Introduction:

Problème : comment saisir l'ensemble des notes de 20 stagiaires dans les modules d'algorithme, de Pascal, de Delphi et d'anglais :

	Algorithme	Pascal	Delphi	anglais
Stagiaire 1	**10**	**11**	**12**	**05**
Stagiaire 2	**12**	**07**	**14**	**10**
….	**11**	**12**	**08**	**11**
Stagiaire 20	**08**	**12**	**11**	**14**

La solution est d'utiliser un tableau à 2 dimensions

Les tableaux à deux dimensions

2) définition: Un tableau à deux dimensions peut être vu comme une collection d'éléments de même nature représentée sous la forme **d'une table constituée de *lignes* et de *colonnes*.**

Les tableaux à deux dimensions sont utilisés généralement dans le calcul des ***matrices***.

❖ **Caractéristiques:**

1. Possède un nom
2. Occupe une zone mémoire contiguë
3. Possède un nombre fini de lignes et de colonnes

❖ **Les cases du tableau**

4. Chaque case du tableau est identifiée par un ***n° de ligne*** et un ***n° de colonne***.
5. Tous les éléments d'un tableau sont du même type.

Les tableaux à deux dimensions

3) Déclaration d'un tableau à deux dimensions :

Pour déclarer un tableau à deux dimensions on utilise la syntaxe suivante :

nomTableau : **tableau** [1.. nbLigne, 1.. nbColonne] **de typeDonnées;**

Exemple :

VAR Notes : tableau [**1..20,1..4**] de **réel** ;

Le tableau nommé Notes contient 20 lignes et 4 colonnes soit 20x4 = 80 cases de type réels.

Les tableaux à deux dimensions

4) Manipulation des tableaux à deux dimension:

- Pour désigner un élément du tableau, on indique le **n° de ligne suivi du n° de colonne**, séparés par une virgule, le tout entre crochets.
- **nomTableau [i,k]** : représente le contenu de la "case" situer à l'intersection de la **$i^{ème}$ ligne et de la $k^{ème}$ colonne**.

➢ Exemple :

Notes [2, 4] représente le contenu de la "case" situer à l'intersection de la $2^{ème}$ ligne et de la $4^{ème}$ colonne. Autrement dit, la note du stagiaire 2 dans le module anglais.

Les tableaux à deux dimensions

- Les éléments d'un tableau à 2 dimensions peuvent être utilisés comme n'importe quelle autre variable classique : affectation, expression arithmétique, comparaison, affichage, saisie.
- La valeur des indices doit être entière et comprise entre les valeurs minimales et maximales déterminées à la déclaration du tableau.

Les tableaux à deux dimensions

- **Exemple :**

  Soit le tableau déclaré ainsi

  VAR

  tab : tableau [1..20, 1..12] d'entiers ;

  Il est impossible d'écrire

  tab[**0**, 5]

  tab[3, **15**]

  tab[**16**, 0]

  tab[**34**, 2]

  Ces éléments n'existent pas

Les tableaux à deux dimensions

- Les éléments constituant un tableau sont indicés : cela permet de parcourir tous les éléments en faisant varier les deux indices du tableau à l'aide de deux boucles imbriquées :
  - Une boucle **POUR** si l'on connait le nombre d'éléments à parcourir
  - Une boucle **TANT QUE** ou **REPETER** si on ne le connait pas

Les tableaux à deux dimensions

5) Lecture des éléments d'une Matrice: On utilise le traitement suivant pour la lecture des éléments d'une Matrice :

```
Algorithme lecture ;
VAR
    m, n : entier;
    Matrice: tableau [1..3, 1..30] de réels ;
Début
  POUR m de 1 à 3 FAIRE
          POUR n de 1 à 30 FAIRE
                  Lire (Matrice [m, n]) ;
          FINPOUR;
  FINPOUR;
FIN.
```

Les tableaux à deux dimensions

5) l'affichage des éléments d'une Matrice (Ecriture) :

On utilise le traitement suivant pour l'écriture des éléments d'une Matrice :

```
Algorithme affiche ;
VAR
    m, n : entier;
    Matrice: tableau [1..3, 1..30] de réels ;
Début
    POUR m de 1 à 3 FAIRE
        POUR n de 1 à 30 FAIRE
            Ecrire (Matrice [m, n]) ;
        FINPOUR;
    FINPOUR;
FIN.
```

Les tableaux à deux dimensions

Exercice : saisir les éléments d'une matrice M de 50 lignes et 100 Colonnes.(nécessitant deux boucles).

```
Algorithme  Saisie;
VAR
        m,  n : entier;
        M: tableau [1..50, 1..100] de réels ;
Début
        Pour i := 1 à 50 faire
                pour j:= 1 à 100 faire
                        lire (M[i,j]);
                Fp;
        Fp;
Fin.
```

Fin du cours

QUESTIONS?