Programmation et Structure de Données

Structure de données de Base
Les Tableaux (partie 1)

ENS KATEB

Les Tableaux à une dimension

Écrire un programme qui permet de lire les notes des étudiants de la section B puis affiche la note maximale et la note minimale.

PB : si on a 200 étudiants on doit donc déclarer 200 variables!!!

Sol : **Rassembler toutes ces variables en une seule (utilisation des tableaux)**

- **Déf : Un ensemble de valeurs portant le même nom de variable et repérées par un nombre, s'appelle un tableau (vecteur).**

  Le nombre qui sert à repérer chaque valeur s'appelle l'indice.

On peut déclarer une seule variable qui peut représenter 200 valeurs possibles

Les Tableaux à une dimension

Déclaration :

Nom_Tab : Tableau [dim] de type; où dim est la taille du tableau et doit être expression constante.

Ex : **SectB : Tableau [200] de Réel;**

Donc SectB est un tableau qui contient 200 élements de type réel.

Pour utiliser les élements de notre tableau on doit accéder par l'indice qui est un **entier positif** et varie entre 0 et dim-1

Ex : dans SectB l'indice varie entre 0 et 199.

Si on veut lire l'élément 0 : lire(SectB[0]);

Attention : ne jamais utiliser un tableau comme une variable simple. Ex : lire(SectB); ou SectB ← 13,5;

Les Tableaux à une dimension

Règles:

1. **Les éléments du tableau** : traités comme des valeurs qui peuvent apparaitre à gauche ou à droite d'une affectation
   ex : T[2] ← X/2; Y ← T[5]- T[1]*0,05; …
2. **Les indices** : sont des entiers positifs qui peuvent être des variables uniques ou des expressions mais qui doivent retourner des entiers positifs
   ex : i,j : entier;
   les écritures suivantes T[i+j], T[2*i+3*j], T[i mod 2], T[2*i-3*j], T[5*i div j], … sont correctes.
   mais T[i/3], T[5,4] sont erronées.
3. **La dimension** : doit être une constante ou une expression constante avec une valeur strictement positive.
   ex : const N ← 100, M ← 50;
   var K: entier;
   T : tableau [N] de réel; ✅
   T : tableau [M*N] de réel; ✅
   T : tableau [10*N + 3*M] de réel; ✅
   T : tableau [K] de réel; ❌
   …

Les Tableaux à une dimension

Utilisation des tableaux:

Pour utiliser les tableaux (lecture, écriture, affectations, …) on utilise souvent les boucles et des constantes pour leur dimensions

Ex : écrire un Algo qui déclare un tableau de 20 entiers et mettre ses éléments à 0 puis affiche son contenu.

```
Algo Tab
const N ← 20;
Var i : entier;
T : Tableau[N] d'entier;
Début
   pour (i de 0 à N-1) faire
      T[i] ← 0;
   FinP;
   pour (i de 0 à N-1) faire
      écrire (T[i], ''\n'');
   FinP;
Fin.
```

OU

```
Algo Tab
const N ← 20;
Var i : entier;
T : Tableau[N] d'entier;
Début
   pour (i de 0 à N-1) faire
      T[i] ← 0;
      écrire (T[i], ''\n'');
   FinP;
Fin.
```

Les Tableaux à une dimension

En C++:

Déclaration :

Type nom_tab[dim]; ex : int A[10];

Initialisation :

ex : int tab[ ] = {1, 20, 15, 6, 18};

La dimension est définit par le nombre des éléments, ici = 5

ex : int tab[5] = {1, 3, 8, 5, 9}

Où chaque élément du tableau est initialisé par la valeur correspondante

ex : int tab[5] = {10, 20} ;

int tab[5] = {10, 20, 5} ;

On peut initialiser seulement les premières valeurs

Utilisation :

```
float tab[5];  int i;
for (i=0 ; i<5 ; i++)
  { cout << "donnez la note numéro " << i+1 << " : " ;
    cin >> tab[i] ;
  }
```

Les Tableaux à deux dimensions

Les tableaux à deux dimensions s'appellent aussi des matrices

Déclaration :

Nom_Mat : Tableau [dimL][dimC] de type;

où dimL représente le nombre des lignes et dimC représente le nombre des colonnes.

ex : **Mat : Tableau[5][3] de entier**; une matrice de 5 lignes et 3 colonnes

Utilisation :

Pour utiliser les matrices il faut deux indices, le 1ere pour indiquer la ligne et le 2eme pour indiquer la colonne.

Ex : M[1][2] ← 20; affecter la valeur 20 à l'élément se trouvant au croisement de la ligne 1 avec la colonne 2

- L'indice de lignes se varie entre 0 et dimL-1
- L'indice de colonne se varie entre 0 et dimC-1

Les règles sur les tableaux à une dimension s'applique sur les matrices

Les Tableaux à deux dimensions

Pour utiliser les matrices (lecture, écriture, affectations, …) on utilise une boucle dans une autre boucle (imbrication des boucles)

Ex : écrire un algorithme qui permet de déclarer un tableau M (4x3) d'entiers, puis demande à l'utilisateur de saisir les éléments en lui indiquant à chaque fois la ligne et la colonne de l'élément à lire, ensuite affiche le tableau sous forme d'une matrice.

Ex :

donner M[0,0] : 3
donner M[0,1] : 9
donner M[0,2] : 2
donner M[1,0] : 5
donner M[1,1] : 8
donner M[1,2] : 2
donner M[2,0] : 5
donner M[2,1] : 1
donner M[2,2] : 4
donner M[3,0] : 6
donner M[3,1] : 7
donner M[3,2] : 3

Affichage :

3	9	2
5	8	2
5	1	4
6	7	3

Les Tableaux à deux dimensions

Solution :

```
Algo MAT
const L ← 4, C ← 3;
Var i,j : entier;
M : Tableau[L][C] d'entier;
Début
   écrire ("Lecture du tableau:\n");
   pour (i de 0 à L-1) faire
        pour (j de 0 à C-1) faire
           écrire ("Donner l'élément
                     M[" , i ,  "," , j, " ] :");
           lire (M[i][j]);
        FinP;
   FinP;
```

(suite)

```
   écrire ("Affichage du tableau:\n ");
   pour (i de 0 à L-1) faire
      pour (j de 0 à C-1) faire
          écrire (M[i][j], "     ");
      FinP;
      écrire ("\n");
    FinP;
FIN.
```

Les Tableaux à deux dimensions

En C++:

Déclaration :

Type nom_tab[dimL][dimC]; ex : int M[3][4];

Initialisation : tous les éléments :

- int tab [3] [4] = { {1, 2, 3, 4} , {5, 6, 7, 8} , {9,10,11,12} }
- int tab [3] [4] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12} ;

Les premiers éléments :

- int tab [3] [4] = { {1, 2} , {3, 4, 5} } ;
- int tab [3] [4] = {1, 2 , 3, 4, 5} ;

NB : les deux dernières initialisations ne sont pas équivalentes

Utilisation :

```
for (i=0 ; i<3 ; i++)
  for (j=0 ; j<4 ; j++)
  { cout << "donnez l'element " << i << " , " << j << " : ";
    cin >> tab[i][j] ;
  }
```

(démonstration C++)