Partie 6
Les Tableaux – les Listes – les Uplets

Utilité des tableaux

Imaginons que dans un programme, nous ayons besoin simultanément de 12 valeurs (par exemple, vos notes pour calculer une moyenne). On pourrait déclarer douze variables, appelées par exemple Notea, Noteb, Notec, etc. On peut opter pour une notation simplifiée, par exemple N1, N2, N3, etc. Mais cela ne change pas notre problème, car arrivé au calcul, et après une succession de douze instructions « Lire » distinctes, cela donnera obligatoirement une atrocité du genre :
Moy ← (N1+N2+N3+N4+N5+N6+N7+N8+N9+N10+N11+N12)/12
Imaginez maintenant que vous vouliez faire un programme qui calcule la moyenne des températures de toutes les journées des 10 dernières années et on comprend que cette solution est inenvisageable.

C'est pourquoi la programmation nous permet de rassembler toutes ces variables en une seule, au sein de laquelle chaque valeur sera désignée par un numéro. En français, cela donnerait donc quelque chose du genre « la note numéro 1 », « la note numéro 2 », « la note numéro 8 ». C'est largement plus pratique.

Notation et utilisation algorithmique

Dans notre exemple, nous créerons donc un tableau appelé Note. Chaque note individuelle (chaque élément du tableau Note) sera donc désignée Note[0], Note[1], etc.

Un tableau doit être déclaré comme tel, en précisant le nombre et le type de valeurs qu'il contiendra.

En nous calquant sur les choix les plus fréquents dans les langages de programmations, nous déciderons ici arbitrairement que :

- les "cases" sont numérotées à partir de zéro, autrement dit que le plus petit indice est zéro.
- lors de la déclaration d'un tableau, on précise le nombre de cases du tableau

Tableau Note[12] en Entier

On peut créer des tableaux contenant des variables de tous types : tableaux de numériques, bien sûr, mais aussi tableaux de caractères, tableaux de booléens, tableaux de tout ce qui existe dans un langage donné comme type de variables. Par contre, seul quelques langages autorisent de faire un mixage de types différents de valeurs au sein d'un même tableau. L'énorme avantage des tableaux, c'est qu'on va pouvoir les traiter en faisant des boucles. Par exemple, pour effectuer notre calcul de moyenne, cela donnera par exemple :

```
Tableau Note[12] en Numérique
Variables Moy, Som en Numérique
Début
Pour i ← 0 à 11
   Ecrire "Entrez la note n°", i
   Lire Note[i]
i Suivant
Som ← 0
Pour i ← 0 à 11
   Som ← Som + Note(i)
i Suivant
Moy ← Som / 12
Fin
```

Remarque : l'indice qui sert à désigner les éléments d'un tableau peut être exprimé directement comme un nombre, mais il peut être aussi une variable, ou une expression calculée

.Exercice 6.1 Ecrire un algorithme qui déclare et remplisse un tableau de 7 valeurs numériques en les mettant toutes à zéro.

Exercice 6.2 Ecrire un algorithme qui déclare et remplisse un tableau contenant les six voyelles de l'alphabet.

Exercice 6.3 Ecrire un algorithme qui déclare un tableau de 9 notes, dont on fait ensuite saisir les valeurs par l'utilisateur.

Exercice 6.4 Que produit l'algorithme suivant ?

```
Tableau Nb(5) en Entier
Variable i en Entier
Début
Pour i ← 0 à 5
  Nb(i) ← i * i
i suivant
Pour i ← 0 à 5
  Ecrire Nb(i)
i suivant
Fin
```

Peut-on simplifier cet algorithme avec le même résultat ?

```
Tableau Nb(5) en Numérique
Variable i en Numérique
Début
Pour i ← 0 à 5
  Nb(i) ← i * i
  Ecrire Nb(i)
i Suivant
Fin
```

Exercice 6.5 Que produit l'algorithme suivant ?

```
Tableau N(6) en Entier
Variables i, k en Entier
Début
N(0) ← 1
Pour k ← 1 à 6
  N(k) ← N(k-1) + 2
k Suivant
Pour i ← 0 à 6
  Ecrire N(i)
i suivant
Fin
```

Peut-on simplifier cet algorithme avec le même résultat ?

```
Tableau N(6) en Numérique
Variables i, k en Numérique
Début
N(0) ← 1
Ecrire N(0)
Pour k ← 1 à 6
  N(k) ← N(k-1) + 2
  Ecrire N(k)
k Suivant
Fin
```

Exercice 6.6 : Que produit l'algorithme suivant ?

```
Tableau Suite(7) en Entier
Variable i en Entier
Début
Suite(0) ← 1
Suite(1) ← 1
Pour i ← 2 à 7
  Suite(i) ← Suite(i-1) + Suite(i-2)
i suivant
Pour i ← 0 à 7
  Ecrire Suite(i)
i suivant
Fin
```

Exercice 6.7 Ecrivez la fin de l'algorithme 6.3 afin que le calcul de la moyenne des notes soit effectué et affiché à l'écran.

3. Tableaux dynamiques

Il arrive fréquemment que l'on ne connaisse pas à l'avance le nombre d'éléments que devra comporter un tableau. Bien sûr, une solution consisterait à déclarer un tableau gigantesque pour être sûr que « ça rentre ». Mais d'une part, on n'en sera jamais parfaitement sûr, d'autre part, en raison de l'immensité de la place mémoire réservée – et la plupart du temps non utilisée.
Aussi, pour parer à ce genre de situation, on a la possibilité de déclarer le tableau sans préciser au départ son nombre d'éléments. Ce n'est que dans un second temps, au cours du programme, que l'on va fixer ce nombre via une instruction de redimensionnement : Redim.
Notez que tant qu'on n'a pas précisé le nombre d'éléments d'un tableau, d'une manière ou d'une autre, ce tableau est inutilisable.

Exemple : on veut faire saisir des notes pour un calcul de moyenne, mais on ne sait pas combien il y aura de notes à saisir. Le début de l'algorithme sera quelque chose du genre :

```
Tableau Notes() en Numérique
Variable nb en Numérique
Début
Ecrire "Combien y a-t-il de notes à saisir ?"
Lire nb
Redim Notes(nb-1)
…
```

Exercice 6.8

Ecrivez un algorithme permettant à l'utilisateur de saisir un nombre quelconque de valeurs, qui devront être stockées dans un tableau. L'utilisateur doit donc commencer par entrer le nombre de valeurs qu'il compte saisir. Il effectuera ensuite cette saisie. Enfin, une fois la saisie terminée, le programme affichera le nombre de valeurs négatives et le nombre de valeurs positives.

Exercice 6.9 Ecrivez un algorithme calculant la somme des valeurs d'un tableau (on suppose que le tableau a été préalablement saisi).

```
Variables i, Som, N en Numérique
Tableau T() en Numérique
Debut

Redim T(N-1)
...
Som ← 0
Pour i ← 0 à N - 1
  Som ← Som + T(i)
i Suivant
Ecrire "Somme des éléments du tableau : ", Som
Fin
```

Exercice 6.10 Ecrivez un algorithme constituant un tableau, à partir de deux tableaux de même longueur préalablement saisis. Le nouveau tableau sera la somme des éléments des deux tableaux de départ.

Tableau 1 :

4	8	7	9	1	5	4	6

Tableau 2 :

7	6	5	2	1	3	7	4

Tableau à constituer :

11	14	12	11	2	8	11	10

```
Variables i, N en Numérique
Tableaux T1(), T2(), T3() en Numérique
Debut

Redim T3(N-1)
...
Pour i ← 0 à N - 1
  T3(i) ← T1(i) + T2(i)
i Suivant
Fin
```

Exercice 6.11 Toujours à partir de deux tableaux précédemment saisis, écrivez un algorithme qui calcule le schtroumpf des deux tableaux. Pour calculer le schtroumpf, il faut multiplier chaque élément du tableau 1 par chaque élément du tableau 2, et additionner le tout. Par exemple si l'on a :

Tableau 1 :

4	8	7	12

Tableau 2 :

3	6

Le Schtroumpf sera :

3 * 4 + 3 * 8 + 3 * 7 + 3 * 12 + 6 * 4 + 6 * 8 + 6 * 7 + 6 * 12 = 279

Exercice 6.12 Ecrivez un algorithme qui permette la saisie d'un nombre quelconque de valeurs, sur le principe de l'ex 6.8. Toutes les valeurs doivent être ensuite augmentées de 1, et le nouveau tableau sera affiché à l'écran.

Exercice 6.13 Ecrivez un algorithme permettant, toujours sur le même principe, à l'utilisateur de saisir un nombre déterminé de valeurs. Le programme, une fois la saisie terminée, renvoie la plus grande valeur en précisant quelle position elle occupe dans le tableau. On prendra soin d'effectuer la saisie dans un premier temps, et la recherche de la plus grande valeur du tableau dans un second temps.

Exercice 6.14 Toujours et encore sur le même principe, écrivez un algorithme permettant, à l'utilisateur de saisir les notes d'une classe. Le programme, une fois la saisie terminée, renvoie le nombre de ces notes supérieures à la moyenne de la classe.

Tableaux Multidimensionnels

Tableaux à deux dimensions

L'informatique nous offre la possibilité de déclarer des tableaux dans lesquels les valeurs ne sont pas repérées par une seule, mais par **deux coordonnées**.
Un tel tableau se déclare ainsi :

Tableau Cases[7, 7] en Numérique

REMARQUE :
*Il n'y a aucune différence qualitative entre un tableau à deux dimensions (i, j) et un tableau à une dimension (i * j). Tout problème qui peut être modélisé d'une manière peut aussi être modélisé de l'autre. Simplement, l'une ou l'autre de ces techniques correspond plus spontanément à tel ou tel problème, et facilite donc (ou complique, si on a choisi la mauvaise option) l'écriture et la lisibilité de l'algorithme.*

Une question classique à propos des tableaux à deux dimensions est de savoir si le premier indice représente les lignes ou le deuxième les colonnes, ou l'inverse. Cette question **n'a aucun sens**. « Lignes » et « Colonnes » sont des concepts graphiques, visuels, qui s'appliquent à des objets du monde réel ; les indices des tableaux ne sont que des coordonnées logiques, pointant sur des adresses de mémoire.

Tableaux à n dimensions

Si vous avez compris le principe des tableaux à deux dimensions, il n'y a aucun problème à passer au maniement de tableaux à trois, quatre, neuf dimensions. C'est exactement la même chose. Si je déclare un tableau Titi[2, 4, 3, 3], il s'agit d'un espace mémoire contenant 3 x 5 x 4 x 4 = 240 valeurs. Chaque valeur y est repérée par quatre coordonnées.
Le principal obstacle au maniement de ces tableaux à plus de trois dimensions est que le programmeur, quand il conçoit son algorithme, aime bien imaginer les boucles dans sa tête, ou faire un petit dessin. Or, autant il est facile d'imaginer concrètement un tableau à une dimension, autant cela reste faisable pour deux dimensions, autant cela devient l'apanage d'une minorité privilégiée pour les tableaux à trois dimensions et hors de portée de tout mortel au-delà. Donc, pour des raisons uniquement pratiques, les tableaux à plus de trois dimensions sont rarement utilisés par des programmeurs .

Exercice 8.1 Écrivez un algorithme remplissant un tableau de 6 sur 13, avec des zéros.

Exercice 8.2

Quel résultat produira cet algorithme ?

```
Tableau X(1, 2) en Entier
Variables i, j, val en Entier
Début
Val ← 1
Pour i ← 0 à 1
  Pour j ← 0 à 2
    X(i, j) ← Val
    Val ← Val + 1
  j Suivant
i Suivant
Pour i ← 0 à 1
  Pour j ← 0 à 2
    Ecrire X(i, j)
  j Suivant
i Suivant
Fin
```

Exercice 8.3

Quel résultat produira cet algorithme ?

```
Tableau X(1, 2) en Entier
Variables i, j, val en Entier
Début
Val ← 1
Pour i ← 0 à 1
  Pour j ← 0 à 2
    X(i, j) ← Val
    Val ← Val + 1
  j Suivant
i Suivant
Pour j ← 0 à 2
  Pour i ← 0 à 1
    Ecrire X(i, j)
  i Suivant
j Suivant
Fin
```

Exercice 8.4

Quel résultat produira cet algorithme ?

```
Tableau T(3, 1) en Entier
Variables k, m, en Entier
Début
Pour k ← 0 à 3
  Pour m ← 0 à 1
    T(k, m) ← k + m
  m Suivant
k Suivant
Pour k ← 0 à 3
  Pour m ← 0 à 1
    Ecrire T(k, m)
  m Suivant
k Suivant
Fin
```

Exercice 8.6 Soit un tableau T à deux dimensions (12, 8) préalablement rempli de valeurs numériques. Écrire un algorithme qui recherche la plus grande valeur au sein de ce tableau.

Les listes :

Les listes ont l'immense avantage sur les tableaux de ne pas avoir une taille fixe et d'accepter de lister des variables de types différents. A part à utiliser ces avantages, les listes s'utilisent de la même manière que les tableaux.

Liste Note(12)

Dans beaucoup de langage, l'utilisation des listes n'est pas facile. En Python par contre, c'est tellement simple qu'on n'utilisera jamais de tableau ! Les listes Python possèdent un tas de methodes :

```
>>> nombres = [17, 38, 10, 25, 72]

>>> nombres.sort()          # trier la liste
>>> nombres
[10, 17, 25, 38, 72]

>>> nombres.append(12)      # ajouter un élément à la fin
>>> nombres
[10, 17, 25, 38, 72, 12]

>>> nombres.reverse()       # inverser l'ordre des éléments
```

```
>>> nombres
[12, 72, 38, 25, 17, 10]

>>> nombres.index(17)          # retrouver l'index d'un élément
4
>>> nombres.remove(38)         # enlever (effacer) un élément
>>> nombres
[12, 72, 25, 17, 10]
```

Exercice 6.15

Ecrivez un algorithme permettant à l'utilisateur de saisir un nombre quelconque de valeurs, qui devront être stockées dans un tableau. L'utilisateur doit donc commencer par entrer le nombre de valeurs qu'il compte saisir. Il effectuera ensuite cette saisie. Enfin, une fois la saisie terminée, le programme affichera le nombre de valeurs négatives et le nombre de valeurs positives.

Les n-uplets (tuples) :

Un tuple (n-uplet) est une liste non-mutable. Un fois créé, **un tuple ne peut en aucune manière être modifié**.

Les éléments d'un tuple ont un ordre défini, tout comme ceux d'une liste. Les indices de tuples débutent à zéro, tout comme ceux d'une liste, le premier élément d'un tuple non vide est toujours t[0].

à quoi servent les tuples ?
Les tuples sont plus rapides que les listes. Si vous définissez un ensemble constant de valeurs et que tout ce que vous allez faire est le parcourir, utilisez un tuple au lieu d'une liste.
Votre code est plus sûr si vous «protégez en écriture» les données qui n'ont pas besoin d'être modifiées. Utiliser un tuple à la place d'une liste revient à avoir une assertion implicite que les données sont constantes et que des mesures spécifiques sont nécessaires pour modifier cette définition.