Les tableaux

Pourquoi les tableaux ?

Imaginons que l'on veuille calculer la moyenne des notes d'une promotion, quel algorithme allons nous utiliser ?
Pour l'instant on pourrait avoir l'algorithme suivant :

```
Algorithme moyenne {Affichage de la moyenne des notes d'une promo saisies par le prof}
{déclarations : réservation d'espace mémoire}
variables      somme, nbEleves, uneNote, i: naturels
début
        {préparation du traitement}
        somme ← 0.0 {initialisation du total à 0 avant le cumul}
        afficher("Nombre d'élèves ?")
        saisir(nbEleves)
        pour i ← 0 à nbEleves faire
                afficher("Saisissez une note :")
                saisir(uneNote)
                somme ← somme + uneNote {cumul}
        fpour
        afficher("Moyenne des notes ", somme/nbEleves)
fin
```

Imaginons que l'on veuille toujours calculer la moyenne des notes d'une promotion mais en gardant en mémoire toutes les notes des étudiants (pour par exemple faire d'autres calculs tels que l'écart type, la note minimale, la note maximale, etc.).
Il faudrait alors déclarer autant de variables qu'il y a d'étudiants, par exemple en supposant qu'il y ait 3 étudiants, on aurait l'algorithme suivant :

```
procedure moyenne()
{déclarations : réservation d'espace mémoire}
variables      somme, note1, note2, note3: naturels
début
        {préparation du traitement}
        somme ← 0.0 {initialisation du total à 0 avant le cumul}
        afficher("Saisissez successivement les notes des 3 étudiants :")
        saisir(note1)
        saisir(note2)
        saisir(note3)
        somme ← note1 + note2 + note3 {cumul}
        afficher("Moyenne des notes ", somme/3)
fin
```

Le problème est que cet algorithme ne fonctionne que pour 3 étudiants. Si on en a 10, il faut déclarer 10 variables. Si on en a n, il faut déclarer n variables . . . ce n'est pas réaliste.
Il faudrait pouvoir, par l'intermédiaire d'une seule variable, stocker plusieurs valeurs de même type … c'est le rôle des tableaux.

Les tableaux à une dimension

C'est ce que l'on nomme un type complexe (en opposition aux types simples vus précédemment).
Le type défini par un tableau est fonction :

- du nombre d'éléments maximal que peut contenir le tableau,
- du type des éléments que peut contenir le tableau.

Par exemple un tableau d'entiers de taille 10 et un tableau d'entiers de taille 20 sont deux types différents.
On peut utiliser directement des variables de type tableau, ou définir de nouveau type à partir du type tableau.
On utilise un type tableau via la syntaxe suivante :

> **tableau**[*intervalle*] **de** <*type des éléments stockés par le tableau*> où intervalle est un intervalle sur un type simple dénombrable avec des bornes constantes.

Par exemple : **type** notes = **tableau**[1 … 26] **de** naturels
défini un nouveau type appelé "notes", qui est un tableau de 26 naturels.

On accède (en lecture ou en écriture) à la $i^{ème}$ valeur d'un tableau en utilisant la syntaxe suivante : <*nom de la variable*>*[indice]*.
Par exemple si "tab" est un tableau de 10 entiers (tab : **tableau**[1 ...10] **d'**entiers), *tab[2]* ← *-5* met la valeur "-5" dans la $2^{ème}$ case du tableau.
En considérant le cas où *a* est une variable de type entier, a ← tab[2] met la valeur de la $2^{ème}$ case du tableau "tab" dans "a", c'est-à-dire -5.

```
Algorithme moyenne {Affichage de la moyenne des notes d'une promo saisies par le prof}
{déclarations : réservation d'espace mémoire}
variables     somme, nbEleves, i: naturels
              notes : tableau[1 … 100] de naturels
début
    {préparation du traitement}
    somme ← 0.0 {initialisation du total à 0 avant le cumul}
    répéter
        afficher("Nombre d'élèves (max. 100) ?")
        saisir(nbEleves)
    tant que nbEleves ≤ 0 et nbEleves > 100
    pour i ← 1 à nbEleves faire
        afficher("Saisissez une note :")
        saisir(notes[i])
        somme ← somme + notes[i] {cumul}
    fpour
    afficher("Moyenne des notes ", somme/nbEleves)
    pour i ← 1 à nbEleves faire
        afficher(notes[i])
    fpour
fin
```

Un tableau possède un nombre maximal d'éléments défini lors de l'écriture de l'algorithme (les bornes sont des constantes explicites, par exemple 10, ou implicites, par exemple MAX). Ce nombre d'éléments ne peut être fonction d'une variable.
Par défaut si aucune initialisation n'a été effectuée les cases d'un tableau possèdent des valeurs aléatoires.
Le nombre d'éléments maximal d'un tableau est différent du nombre d'éléments significatifs dans un tableau. Dans l'exemple précédent le nombre maximal d'éléments est de 100 mais le nombre significatif d'éléments est référencé par la variable "nbEleves".
L'accès aux éléments d'un tableau est direct (temps d'accès constant).
Il n'y a pas conservation de l'information d'une exécution du programme à une autre.

Les tableaux à deux dimensions

On peut aussi avoir des tableaux à deux dimensions (permettant ainsi de représenter par exemple des matrices à deux dimensions).
On déclare une matrice à deux dimensions de la façon suivante :
tableau[*intervallePremièreDimension*][*intervalleDeuxièmeDimension*] **de** <type des éléments>

On accède (en lecture ou en écriture) à la $i^{ème}$, $j^{ème}$ valeur d'un tableau en utilisant la syntaxe suivante : *<nom de la variable>[i][j]*

Par exemple si "tab" est défini par tab : **tableau**[1 … 3][1 … 2] **de** réels, *tab[2][1]* ← *-1.2* met la valeur "-1.2" dans la case "2,1" du tableau.
En considérant le cas où "a" est une variable de type réel, *a* ← *tab[2][1]* met "-1.2" dans "a".

Les tableaux à n dimensions

Par extension, on peut aussi utiliser des tableaux à plus grande dimension.
Leur déclaration est à l'image des tableaux à deux dimensions, c'est-à-dire :
tableau[*intervalle1*][*intervalle2*]...[*intervallen*] **de** <type des valeurs>

Par exemple si "tab" est défini par tab : **tableau**[1 … 10][0 … 9]['a' … 'e'] **d'**entiers, *tab[2][1]['b']* ←*10* met la valeur "10" dans la case "2, 1, 'b'" du tableau.
En considérant le cas où "a" est une variable de type réel, *a* ←*tab[2][1]['b']* met "10" dans "a".

Mise en pratique

Exercice 0

Écrire un programme qui affiche en ordre croissant les notes d'une promotion de 10 élèves, suivies de la note la plus faible, de la note la plus élevée et de la moyenne.