Les Tableaux

http://pise.info/algo/tableaux.htm

« Si on ment à un compilateur, il prendra sa revanche. » - Henry Spencer.

Utilité des tableaux

Imaginons que dans un programme, nous ayons besoin simultanément de 12 valeurs (par exemple, des notes pour calculer une moyenne). Évidemment, la seule solution dont nous disposons à l'heure actuelle consiste à déclarer douze variables, appelées par exemple `noteA`, `noteB`, `noteC`, etc. Bien sûr, on peut opter pour une notation un peu simplifiée, par exemple `n1`, `n2`, `n3`, etc. Mais cela ne change pas fondamentalement 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;
```

Ouf ! C'est tout de même bigrement laborieux. Et pour un peu que nous soyons dans un programme de gestion avec quelques centaines ou quelques milliers de valeurs à traiter, alors là c'est le suicide direct.

Cerise sur le gâteau, si en plus on est dans une situation on l'on ne peut pas savoir d'avance combien il y aura de valeurs à traiter, là on est carrément cuits.

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 bon 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, vous vous en doutez.

Un ensemble de valeurs portant le même nom de variable et repérées par un nombre, s'appelle un tableau, ou encore une variable indicée.

Le nombre qui, au sein d'un tableau, sert à repérer chaque valeur s'appelle – ô surprise – l'indice.

Chaque fois que l'on doit désigner un élément du tableau, on fait figurer le nom du tableau, suivi de l'indice de l'élément, entre parenthèses.

Notation et utilisation algorithmique

Dans notre exemple, nous créerons donc un tableau appelé `notes`. Chaque note individuelle (chaque élément du tableau `notes`) sera donc désignée `notes[0]`, `notes[1]`, etc. **Eh oui, attention, les indices des tableaux commencent généralement à 0, et non à 1**.

Un tableau doit être déclaré comme tel, en précisant le nombre et le type de valeurs qu'il contiendra (la déclaration des tableaux est susceptible de varier d'un langage à l'autre. Certains langages réclament le nombre d'éléments, d'autre le plus grand indice... C'est donc une affaire de conventions).

En nous calquant sur les choix les plus fréquents dans les langages de programmations, nous déciderons ici arbitrairement et une bonne fois pour toutes 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 la plus grande valeur de l'indice (différente, donc, du nombre de cases du tableau, puisque si on veut 12 emplacements, le plus grand indice sera 11). Au début, ça déroute, mais vous verrez, avec le temps, on se fait à tout, même au pire.

```
int[] notes = new int[12];
```

ou

```
int[] notes; =
notes = new int[12];
```

ou

```
int[] notes = {12, 56, 20, 1, 78, 1};
```

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, hormis dans quelques rares langages, on ne peut pas 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 :

```
class CalculMoyenne {
    void principal() {
        int[] notes = new int[12];
        int som;
        double moy;
        int i;

        i = 0;
        while (i < 12) {
            notes[i] = SimpleInput.getInt
              ("Entrez la note n° " + i, @notes[i]);
            i = i + 1;
        }

        som = 0;
        i = 0;
        while (i < 12) {
            som = som + notes[i];
            i = i + 1;
        }

        moy = som / 12;
        System.out.println("moyenne = " + moy);
    }
}
```

NB : On a fait deux boucles successives pour plus de lisibilité, mais on aurait tout aussi bien pu n'en écrire qu'une seule dans laquelle on aurait tout fait d'un seul coup.

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

Dans un tableau, la valeur d'un indice doit toujours :

- être égale au moins à 0 (dans quelques rares langages, le premier élément d'un tableau porte l'indice 1). Mais comme je l'ai déjà écrit plus haut, nous avons choisi ici de commencer la numérotation des indices à zéro, comme c'est le cas en langage C et en Visual Basic. Donc attention, `truc[6]` est le septième élément du tableau `truc` !
- être un nombre entier Quel que soit le langage, l'élément `truc[3.1416]` n'existe jamais.
- être inférieure ou égale au nombre d'éléments du tableau (moins 1, si l'on commence la numérotation à zéro). Si le tableau `bidule` a été déclaré comme ayant 25 éléments, la présence dans une ligne, sous une forme ou sous une autre, de `bidule[32]` déclenchera automatiquement une erreur.

Je le re-re-répète, si l'on est dans un langage où les indices commencent à zéro, il faut en tenir compte à la déclaration :

```
int[] notes = new int[14];
```

...créera un tableau de 14 éléments, le plus petit indice étant 0 et le plus grand 13.

LE GAG DE LA JOURNEE

Il consiste à confondre, dans sa tête et/ou dans un algorithme, l'indice d'un élément d'un tableau avec le contenu de cet élément. La troisième maison de la rue n'a pas forcément trois habitants, et la vingtième vingt habitants. En notation algorithmique, il n'y a aucun rapport entre `i` et `truc[i]`.