Algorithmie et Programmation

Chapitre 5 : les tableaux

---

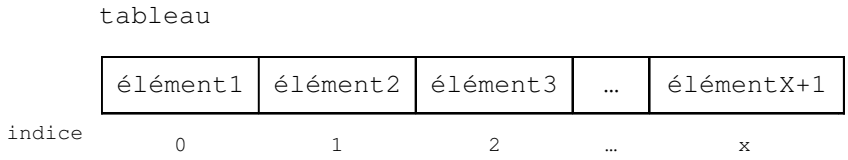
1 De l'utilité des tableaux.

Imaginons que dans un programme, nous ayons besoin simultanément de 8 valeurs (par exemple, des notes pour calculer une moyenne). La seule solution dont nous disposons à l'heure actuelle est un algorithme de gestion de flot. Malheureusement un algorithme de gestion de flot de base ne permet pas de stocker les membres du flot, ce qui peut poser problème si l'on veut réutiliser ces valeurs plus tard. La seule manière serait donc de stocker chaque note dans une variable.

Si nous étions dans un programme complexe avec quelques centaines ou quelques milliers de valeurs à traiter, la création de ces variables prendrait beaucoup trop de temps (et de ligne de code). Heureusement il existe une solution: les tableaux !

Un tableau est une structure de donnée qui nous permet de rassembler un nombre défini d'éléments de même type. Pour retrouver chaque élément on aura recours à son indice.

Pour utiliser une image familière on peut voir un tableau comme étant un bureau dans lequel chaque élément prendra place dans un tiroir. Pour retrouver un élément il nous suffit de connaître le numéro du tiroir dans lequel il se trouve.



Dans la plupart des langages de programmation courants l'indice de la première case d'un tableau est 0.

2 Déclaration d'un tableau

Voici comment créer un tableau de 8 notes

Déclaration et création:

```
double[] tableauNotes = new double[8];
```

La déclaration d'un tableau doit préciser trois informations essentielles : le nom de la variable, la taille et le type des valeurs du tableau.

On peut créer des tableaux contenant des variables de tous types. (Booléens, entiers, réels, caractères, String, etc…) Par contre, on ne peut pas faire un mélange de valeurs de types différents au sein d'un même tableau.

Chaque élément du tableau est accessible grâce à son indice.

Exemple : `tableauNotes[0]`, `tableauNotes[7]` , etc.

La valeur d'un indice doit toujours :
- Etre un nombre entier. l'élément `tableauNotes[3.1415]` n'existe donc pas.
- Le premier élément d'un tableau porte l'indice 0.
- L'indice doit être une valeur entière dans l'intervalle [0 ; taille du tableau - 1].

L'indice qui sert à repérer les éléments du tableau peut être exprimé comme :
- Une constante littérale numérique entière : `tableauNotes[3]`
- Une variable contenant une valeur entière : `tableauNotes[indice]`
- Une expression calculée de résultat entier : `tableauNotes[(borneMax-borneMin)\2]`

Remarques :
- Soit un tableau déclaré comme ayant 8 éléments, la présence dans une instruction, sous une forme ou sous une autre, de `tableauNotes[8]` déclenchera automatiquement une erreur. En effet, l'intervalle des indices autorisé est ici [0;7].
- Ne pas confondre l'indice d'un élément d'un tableau avec son contenu. La troisième maison de la rue n'a pas forcément trois habitants, et la vingtième vingt habitants. Il ne faut donc pas confondre les valeurs `indice` et `tableauNotes[indice]`.

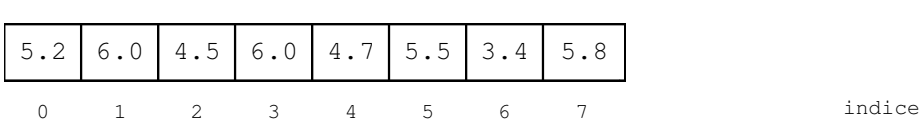
3 Initialisation d'un tableau.

L'exemple précédent n'est pas complet, dans la mesure où aucune initialisation des valeurs du tableau n'a eu lieu. Par défaut Java les initialise à 0 contrairement à d'autre language ou aucune initialisation n'est faites pour vous.

Une initialisation peut se faire de plusieurs manières :

- En donnant dans l'algorithme une valeur à chaque élément :

```
//Affectation de valeurs au tableau
tableauNotes[0] = 5.2
tableauNotes[1] = 6.0
tableauNotes[2] = 4.5
//...
tableauNotes[7] = 5.8
```



Mais cela peut être long à écrire. C'est pourquoi, il existe un raccourci...

- Si on désire un tableau constitué de 8 réels, on peut le déclarer et l'initialiser rapidement par l'instruction suivante :

```
double[] tableauNotes = {5.2, 6.0, 4.5, 6.0, 4.7, 5.5, 3.4, 5.8};
```

4 Algorithme de base sur les tableaux.

Voici une liste non exhaustive d'algorithme de base (à faire évoluer selon le problème rencontré) que vous pouvez utiliser avec les tableaux. Remarque: la plupart utilise l'instruction nomDuTableau.length qui retourne la taille d'un tableau. Pour des questions de facilité, tout les algorithmes travaille avec un tableau d'entier.

Parcours:

```
for (int i=0;i<tab.length;i++){
    //séquence d'instructions sur le membre i du tableau
}
```

Parcours inverse:

```
for (int i=tab.length-1;i>=0;i--){
    //séquence d'instructions sur le membre i du tableau
}
```

Initialisation (à 1 dans cet exemple):

```
for (int i=0;i<tab.length;i++){
    tab[i] = 1;
}
```

Affichage:

```
for (int i=0;i<tab.length;i++){
    System.out.print(tab[i] + " ");
}
```

Parcours avec sélection (ici on ne parcours que les nombres pairs):

```
for (int i=0;i<tab.length;i++){
    if (tab[i]%2==0){
       //séquence d'instructions sur le membre i du tableau
    }
}
```

Mise à jour (ici on double toutes les valeurs du tableau):

```
for (int i=0;i<tab.length;i++){
    tab[i] = 2*tab[i];
}
```

Fonction de recherche de la position (algorithme de base):

```
/**
 * Retourne la 1er position de "n" dans le tableau "array".
 * Retourne -1 si la valeur "n" n'est pas dans le tableau.
 */
static int findPos(int n, int[] array){
    int i=0;
    while (i<array.length && array[i]!=n) i++;
    if (i==array.length) i=-1;
    return i;
}
```