xb

TP-3

B. Hugueney, S. Lefebvre, M. Sellami

<2015-10-07 mer.>

Contents

1 Objectifs pédagogiques **2**
1.1 Algorithmique 2
1.2 Java 3

2 Rappels: tableaux "à deux dimensions" **3**
2.1 Déclaration et instanciation 3
2.2 Initialisation et accès 3

3 Format du rendu **4**

4 Exercices **4**
4.1 Saisie dynamique des éléments d'un tableau (20 minutes) 4
4.2 Moyenne des éléments d'un tableau (20 minutes) 4
4.2.1 Écrire une fonction qui calcule et renvoie la moyenne entière des éléments d'un tableau, afficher le premier indice du tableau pour lequel l'élément est égal à la moyenne (s'il y en a un). 4
4.2.2 Ecrire une fonction qui prend un tableau d'entiers en paramètre, et qui déplace les éléments dans le tableau afin d'obtenir une partition en deux : 4
4.2.3 Bonus: 5
4.3 Tri des éléments d'un tableau (40 mn) 5
4.4 Recherche exhaustive (20 mn) 5
4.5 Dichotomie (60 minutes) 5
4.5.1 Bonus 6

1 Objectifs pédagogiques

1.1 Algorithmique

- Conception d'algorithmes non triviaux manipulant des tableaux d'éléments
- Intorduction de la notion de compléxité

1.2 Java

Déclaration et utilisation de tableaux à deux dimensions.

2 Rappels: tableaux "à deux dimensions"

En fait, le langage Java ne définit que des tableaux à une dimension. Mais un type tableau est un type comme les autres et comme il est possible de définir des tableaux de n'importe quel type, il est donc possible de définir un tableau (à une dimension) de [tableau à une dimension] ce qui permet d'implémenter des tableaux "à deux dimensions". Un tableau à deux dimensions de taille L × C (L lignes et C colonnes)sera donc en fait un tableau uni-dimensionnel de taille L dont chaque élément est un tableau unidimensionnel de taille C.

2.1 Déclaration et instanciation

La déclaration d'un tableau à 2 dimensions se fait de la manière suivante (où typeElt est le type des éléments du tableau):

```
1 typeElt[][] nomTableau;
```

Un tableau déclaré doit ensuite être instancié :

```
1 nomTableau = new typeElt[nombresDelignes][nombreDeColonne];
```

Ces deux opérations peuvent se faire simultanément. Par exemple,

```
1 int[][] tab = new int[3][4];
```

déclare et instancie un tableau à 3 lignes et 4 colonnes.

2.2 Initialisation et accès

L'initialisation consiste à attribuer à chacune des cases du tableau une valeur. L'initialisation peut être effectuée lors de la déclaration d'un tableau, enindiquant la liste des valeurs respectives entre accolades. Par exemple, le code:

```
1 int[][] tab = { {0,1,2,3}, {4,5,6,7}, {8,9,10,11} };
```

définit le tableau suivant :

0	1	2	3
4	5	6	7
8	9	10	11

L'initialisation peut également être réalisée en accédant à chacune des cases du tableau, et en lui attribuant une valeur. L'accès à un élément du tableau se fait en donnant l'indice de la ligne et de la colonne: C

```
nomTableau[indiceLigne][indiceColonne];
```

Ainsi, la case contenant le nombre 6 dans le tableau précédent est donnée par `tab[1][2]` .

3 Format du rendu

Vous déposerez sur moodle un dossier zippé contenant votre projet Eclipse nommé "TP3" ainsi que un document contenant les réponses aux différentes questions du tp.

4 Exercices

4.1 Saisie dynamique des éléments d'un tableau (20 minutes)

- Écrire une fonction qui demande à l'utilisateur d'entrer des valeurs pour remplir un tableau uni-dimensionnel de taille 4, et qui renvoie ce tableau rempli.
- Afficher le tableau une fois la saisie terminée.
- Même question pour un tableau à 3 lignes et 4 colonnes.

4.2 Moyenne des éléments d'un tableau (20 minutes)

4.2.1 Écrire une fonction qui calcule et renvoie la moyenne entière des éléments d'un tableau, afficher le premier indice du tableau pour lequel l'élément est égal à la moyenne (s'il y en a un).

4.2.2 Ecrire une fonction qui prend un tableau d'entiers en paramètre, et qui déplace les éléments dans le tableau afin d'obtenir une partition en deux :

1. les éléments strictement inférieurs à la moyenne
2. les éléments supérieurs ou égaux à la moyenne

exemple de partitionnement

```
2 4 2 9 3 6 7 1 2
moyenne : 4
indice moyenne : 1
```

etape	< 4	$>= 4$
1	2	4 2 9 3 6 7 1 2
2	2 2 2	9 3 6 7 1 4
3	2 2 2 1 3	6 7 9 4

La fonction renvoie le tableau partitionné.

4.2.3 Bonus:

- Réfléchir au problème similaire (?) de la détermination de la médiane. Ecrire la fonction correspondante.
- Concevoir aussi un algorithme permettant la détermination du mode (valeur la plus fréquente dans le tableau) ? Ecrire la fonction correspondante.

4.3 Tri des éléments d'un tableau (40 mn)

Ecrire une fonction qui prend un tableau d'entiers en paramètres, et qui retourne ce tableau trié par ordre croissant.

4.4 Recherche exhaustive (20 mn)

Ecrire une fonction qui prend en paramètre une valeur entière **v** et un tableau d'entiers **t** et qui renvoie l'indice de la première occurence de **v** dans le tableau.

Combien d'itérations sont nécessaires pour trouver v ?

- Si v est au début ?
- Si v est à la fin ?
- en moyenne ?

4.5 Dichotomie (60 minutes)

Implémenter une recherche dichotomique dans un tableau que l'on suppose trié. On rappelle le principe de la recherche dichotomique dans un tableau trié par ordre croissant. En supposant que l'on dispose d'un tableau T dans lequel on cherche l'indice où est stocké l'entier n.

- On tient en permanence deux indices gauche et droite tels que T [gauche] n T [droite].
- À chaque étape, on prend le milieu de l'intervalle [gauche, droite] et on le compare à n.
- En fonction du résultat, soit on affiche l'indice du milieu, soit on met à jour gauche ou droite et on continue à chercher n dans le sous-tableau correspondant.
- On s'arrête lorsque gauche est plus grand que droite : ce cas est un cas d'échec (n n'a été trouvé nulle part) et on affiche l'indice 1.

4.5.1 Bonus

Déterminer combien d'étapes sont nécessaires (en moyenne ? en pire cas ?)
Emacs 24.4.1 (Org mode 8.2.10)