Java

Les Tableaux

TD 4

Exercice 1 :

Ecrire un programme Java qui permet de trouver le nombre d'occurrences d'une valeur entière dans un tableau de 20 valeurs.

Exercice 2 :

Ecrire un programme Java qui demande et stocke N nombres réels positifs dans un tableau T puis en calcule la moyenne et en détermine le maximum et le minimum des éléments du tableau T.

Exercice 4 :

Rédiger un algorithme qui réalise les actions suivantes :

a. demander et lire le nombre d'éléments que le tableau contiendra.
b. remplir le tableau avec des nombres fournis par RANDOM(100)

Exercice 5 :

Rédiger un programme Java qui crée un tableau trié, sans redondance d'éléments, et fasse la recherche d'un élément présent ou absent dans le tableau.

Notes :

1. utilisez N =(int) (Math.random() *51)+1pour déterminer le nombre d'éléments dans le tableau;
2. utilisez T[i+1]= T[i] + (int) (Math.random()*3+1) pour générer les éléments triés du tableau;
3. utilisez X = (int)(Math.random() *30)pour générer l'élément à rechercher dans le tableau;
4. utilisez le fait que le tableau est trié pour accélérer la recherche.

Exercice 6 :

Rédiger un programme Java qui réalise les actions suivantes :

- demander et lire les nombres entiers positifs qui doivent être stockés dans le tableau T. La fin du remplissage du tableau est marquée par l'introduction d'un nombre négatif qui n'appartient pas à T.
- trier le tableau par la méthode du tri par sélection.

Tri par sélection :

Le tri par sélection recherche l'élément le plus petit du tableau et l'échange avec le premier élément. Ensuite, on recommence cette opération avec le sous tableau commençant par le deuxième élément. Le tableau est entièrement trié lorsque le sous tableau ne contient plus qu'un seul sous élément. Ecrire le programme java.

Tri par sélection : illustration

Prenons désormais comme exemple la suite de nombres suivante : 6, 1, 9, 3. Trions cette suite avec l'algorithme du tri par sélection dans l'ordre croissant :

1er tour :
6, 1, 9, 3 → le plus petit élément du tableau est 1, le deuxième élément, on l'échange avec le premier, qui est 6.
2ème tour :
1, 6, 9, 3 →le deuxième plus petit élément du tableau est 3, le quatrième élément, on l'échange avec le deuxième qui est 6.
3ème tour :
1, 3, 9, 6 → le troisième plus petit élément du tableau est 6, le quatrième élément, on l'échange avec le troisième qui est 9.
1, 3, 6, 9

Exemple : Par exemple, pour trier le tableau : **6 2 8 1 5 4**

1 2 8 **6** 5 4
1 **2** 8 6 5 4
1 2 **4** 6 5 **8**
1 2 4 **5** **6** 8
1 2 4 5 **6** 8
1 2 4 5 6 **8**

Série d'exercices en Java : Les tableaux simples

Exercice 1 :

Ecrire un programme Java qui permet de :

a) calculer et afficher la somme des éléments d'un tableau de n éléments.
b) Augmenter de 1 tous les éléments d'un tableau, le nouveau tableau sera affiché à l'écran.
Exemple : tab[4]={1,23,6,9,-1} devient tab[4]={2,24,7,10,0}.
c) d'inverser l'ordre des éléments d'un tableau de N nombres (dans un 2ème tableau).
d) d'inverser l'ordre des éléments d'un tableau de N nombres (dans un même tableau).

Exercice 2 :

Ecrire une méthode Java qui réorganise un tableau d'entiers de telle manière que tous les nombres pairs soient regroupés au début du tableau et suivis par les nombres impairs.

Exercice 3 :

Reprendre le programme du tri par sélection, mais pour un tri décroissant (du plus grand au plus petit).

Exercice 4 :

Ecrire un programme qui saisit un entier au clavier et qui recherche si cet entier appartient au tableau. Au cas où la réponse est positive, l'indice de cet entier dans le tableau est affiché. S'il y a plusieurs occurrences, le dernier indice est affiché

Série d'exercices en Java : Plusieurs tableaux

Exercice 5 :

Écrire un programme Scalaire.java qui calcule le produit scalaire de deux tableaux. Votre programme devra utiliser les éléments suivants :

- Déclarations dans la méthode `main()` :
  - une `variable nMax` représentant la taille maximale des vecteurs (inutile de lui donner une valeur trop élevée... 10 suffit amplement)
  - deux variables `v1` et `v2`, de type tableau de réels de taille `nMax`.
- Méthode :
  - demander à l'utilisateur d'entrer `n`, la taille effective des vecteurs.
  - vérifier que `n` est compris entre 1 et `nMax` et demander à l'utilisateur d'entrer à nouveau une valeur tant que ce n'est pas le cas
  - demander à l'utilisateur les composantes ($v1_0$... $v1_{n-1}$, $v2_0$ ... $v2_{n-1}$) des vecteurs `v1` et `v2`.
  - calculer le produit scalaire de `v1` et `v2`.
  - afficher le résultat.

Note :
Le produit scalaire de *a* par *b* est: $a \cdot b = a_1*b_1 + a_2*b_2 + ... + a_n*b_n$
Exemple : a = (5, 3, -1) b = (2, 1, 2) $a \cdot b = 5*2+3*1+(-1)*2=11$

Exemple d'exécution :
```
Quelle taille pour vos vecteurs [entre 1 et  10] ?
3
Saisie du premier vecteur :
 v1[0] = 5
 v1[1] = 3
 v1[2] = -1
Saisie du second vecteur :
 v2[0] = 2
 v2[1] = 1
 v2[2] = 2
Le produit scalaire de v1 par v2 vaut 11.0
```

Exercice 6 :
Ecrire en Java et tester une fonction permettant de fusionner deux tableaux d'entiers triés par ordre croissant en un seul tableau trié par ordre croissant.

Série d'exercices en Java : Les tableaux + méthodes.

Exercice 7 :
Ecrire en Java une méthode **void affichageTableau (int [] t),** qui affiche le contenu d'un tableau.

Exercice 8 :
Ecrire en Java une méthode **void initialisationTableau(int t[], int max)**qui permet de faire l'initialisation de tous les contenus dans un tableau d'entiers avec une valeur tirée au sort entre 0 et max inclus.

Exercice 9 :
Ecrire en Java une fonction de recherche de l'indice de la valeur minimale d'un intervalle d'un tableau d'entiers compris entre les indices deb et fin inclus.

Exercice 10 :
Ecrire en Java une fonction« **int nbr_occur (int t[]) »**qui recherche et retourne le nombre d'occurrences d'un entier X dans un tableau. X étant un nombre demandé à l'utilisateur. Tester le code avec un tableau d'entiers de taille 30 et contenant des chiffres aléatoires.

Exercice 11 :

Ecrire en Java une méthode supplémentaire **« void test_egalite (int t1[], int t2[]) »** qui permet de tester l'égalité entre les deux tableaux : il affiche VRAI si les composants des deux tableaux correspondent position par position, et FAUX sinon.

Exercice 12 :

Ecrire en Java une fonction**« int test_contenu (int t1[], int t2[]) »** qui vérifie si les 2 tableaux contiennent exactement les mêmes éléments mais pas forcément dans l'ordre : si c'est le cas, la fonction retourne 1 sinon elle retournera 0.

Exercice 13 :

Ecrire en Java une méthode « **void test_exc_prec () »** qui re-remplit un des 2 tableaux jusqu'à obtenir un VRAI de la méthode de l'exercice précédent. Afficher le nombre de tentative ainsi que le tableau.

Exercice 14 :

Ecrire un programme Java qui inverse l'ordre d'apparition des nombres présents dans un tableau d'entiers.