1 - LES TABLEAUX

1. Présentation

Le tableau

Un tableau est un nouveau type de variable qui permet de regrouper dans une seule variable plusieurs valeurs de même type.

Soit « tab » un tableau d'entiers :

- Pour accéder à un élément du tableau, on écrit par exemple : tab[2].
- Le premier élément du tableau c'est tab[1] (tab[0] en langage C et dans les langages dérivés du C)
- tab[1] c'est une variable comme une autre, ici de type entier.

Le taille du tableau

Le nombre de valeurs maximum que peut recevoir un tableau est posé au moment de la définition du tableau.

Ce nombre, c'est la taille du tableau.

C'est un paramètre constitutif du tableau.

Le nombre d'éléments du tableau

Un tableau n'est pas forcément entièrement rempli.

Il faut donc distinguer entre la taille et le nombre d'éléments du tableau à un moment donné.

Ce nombre est nécessaire pour pouvoir parcourir tous les éléments du tableau.

Imbrication de tableaux et d'enregistrements

Tableau d'enregistrement

Enregistrement de tableaux

2. Méthodes de recherche

Recherche dans un tableau non trié

Pour rechercher un élément dans un tableau non trié, il faut parcourir tout le tableau.

Si on a N éléments dans le tableau, la recherche se fera en au maximum N tests, N/2 tests en moyenne.

Recherche dans un tableau trié : la recherche dichotomique

Pour rechercher un élément dans un tableau trié, on fait une recherche dichotomique.

Si on a N éléments dans le tableau, la recherche se fera en au maximum log2(N), soit environ 20 pour N = un million.

3. Méthodes de tri

Tri à bulles

Le principe est de réorganiser (inverser) les couples non classés tant qu'il en existe.

Tri par extraction (élémentaire, heapsort)

L'opération de base consiste à rechercher l'extremum dans la partie non triée et à le permuter avec l'élément frontière, puis à déplacer la frontière d'une position.

Tri par insertion

L'opération de base consiste à prendre l'élément frontière dans la partie non triée et à l'insérer à sa place dans la partie triée, puis à déplacer la frontière d'une position.

Autres méthodes

Il existe de nombreuses autres méthodes qui ne sont pas abordées ici.

4. Exercices sur les tableaux

Complexité

Pour tous les exercices, on donnera la complexité temporelle théorique de l'algorithme proposé.

Tableaux à une dimension

Exercice 1 : somme des éléments d'un tableau

Ecrire une fonction qui calcule la somme des éléments d'un tableau.

Exercice 2 : max d'un tableau

Ecrire une fonction qui détermine la valeur la plus grande d'un tableau. On renverra le maximum puis la position du maximum dans le tableau.

Exercice 3 : décalage ver le bas

Ecrire une fonction qui décale les éléments d'un tableau vers le bas (le $5^{ème}$ passe en $4^{ème}$ par exemple et le premier passe en dernier, le Nième.

Exercice 4 : décalage vers le haut

Ecrire une fonction qui décale les éléments d'un tableau vers le haut (le $4^{ème}$ passe en $5^{ème}$ par exemple et le le dernier, le Nième, passe en premier.

Exercice 5 : inversion d'un tableau

Ecrire une fonction qui inverse un tableau : le premier élément est permuté avec le dernier, le deuxième avec l'avant dernier, etc.

Exercice 6 : recherche d'un élément

Écrire une procédure (ou une fonction) qui recherche une valeur dans un tableau trié sans doublon.

On proposera 2 algorithmes : l'un simplement itératif, l'autre qui utilisera la méthode de recherche dichotomique.

Quelle est la complexité de l'algorithme dans les 2 cas ?

Exercice 7 : tri à bulles

Ecrire l'algorithme de tri d'un tableau par la méthode du tri à bulles.

Le principe est d'inverser les couples de valeurs successives du tableau en fonction de l'ordre du tri, en parcourant le tableau d'un bout à l'autre et en répétant l'opération autant de fois que nécessaire.

Exemple avec le tableau : 6 10 5 8 12 4

On teste 6 et 10 et on ne fait rien

On teste 10 et 5 et on inverse : 6 5 10 8 12 4

On teste 10 et 8 et on inverse : 6 5 8 10 12 4

On teste 10 et 12 et on ne fait rien

On teste 12 et 4 et on inverse : 6 5 8 10 4 12

Le plus grand (12) est tout en bas.

Et on recommence :

On teste 6 et 5 et on inverse : 5 6 10 8 12 4

On teste 6 et 10 et on ne fait rien

Etc.

Quelle est la complexité de l'algorithme ?

Exercice 8 : tri par extraction

Ecrire une fonction qui tri un tableau en utilisant la méthode de tri par extraction. Le principe de ce tri est d'aller chercher le plus petit élément du tableau pour le mettre en premier, puis de recommencer l'opération en partant du second élément du tableau, puis du troisième, etc.

Exercice 9 : suppression d'un élément

Écrire une procédure (ou une fonction) qui supprime une valeur dans un tableau trié sans doublon.

Exercice 10 : ajout d'un élément

Écrire une procédure (ou une fonction) qui ajoute une valeur dans un tableau trié sans doublon.

Exercice 11 : suppression des doublons

Ecrire une fonction qui élimine les doublons d'un tableau d'entiers. On considérera d'abord le tableau comme étant trié puis on traitera le cas général.

Matrices

Exercice 1

Ecrire une fonction d'affichage des éléments d'une matrice. Faites en sorte que l'affichage soit lisible.

Exercice 2

Ecrire une fonction qui calcule la moyenne des sommes des lignes et la moyenne des sommes des colonnes d'une matrice.

Exercice 3

Ecrire une fonction qui calcule la moyenne par ligne et par colonne dans une matrice.

Exercice 4

Ecrire une fonction qui multiplie deux matrices.

Avec A matrice n lignes, m colonnes. B matrice m lignes, p colonnes. C matrice n lignes, p colonnes.

$C_{i,j} = \sum_{k=0}^{m} A_{i,k} * B_{k,j}$.

Exercice 5

Ecrire une procédure qui permet de trier une matrice selon une colonne donnée (utiliser la méthode du tri à bulles).

Exercice 6

Ecrire une procédure qui permet de trier une matrice selon une ligne donnée (utiliser la méthode du tri à bulles).

Exercice 7 : Tours de Hanoi (avec matrice ou avec tableaux)

Ecrire un programme qui affiche les déplacements dans des tours de Hanoï.

Le principe de l'algorithme est le suivant :

On déplace le plus petit à droite

On déplace ce qu'on peut

On recommence jusqu'à ce que ce soit fini.

Pour traiter le problème, il faut se doter d'une matrice de nbLignes et 3 colonnes qu'on appellera « tours ».

L'algorithme du programme est le suivant :

```
Afficher (tours, nblignes)
Tant que vrai
    Déplacer_le_1 (tours, nbLignes)
    Afficher (tours, nblignes)
    Si (fini(tours, nbLignes)) break
    Afficher (tours, nblignes)
    Déplacer_un_autre_que_le_un (tours, nbLignes)
Fin tant que
```

On pourra aussi faire une variante qui affiche des éléments avec des « * », 1 étoile pour le plus petit, 3 étoiles pour le suivant, 5 étoiles ensuite, etc.

On peut aussi faire une variante récursive, l'algorithme étant le suivant :

L'algorithme de la fonction récursive est le suivant :

```
hanoi (tours, nblignes, est_ce_le_1)
    Si (fini(tours, nbLignes)) return ;
    si est_ce_le_1 == 1
        Déplacer_le_1 (tours, nbLignes)
        est_ce_le_1 = 0
    sinon
        Déplacer_un_autre_que_le_un (tours, nbLignes)
        est_ce_le_1 = 1
    finsi
    Afficher (tours, nblignes)
    hanoi (tours, nblignes, est_ce_le_1)
fin
```

L'algorithme du programme est le suivant :

```
Afficher (tours, nblignes)
hanoi (tours, nblignes, 1)
```