POLYTECH PARIS-SUD

Et2-Info

Algorithmique

Travaux Dirigés N° 2

(Correction)

Les algorithmes de ce TD sont à écrire en pseudocode.

1 Tri à bulle

Vous allez écrire un programme relativement connu : le tri à bulle. Dans cet algorithme, on compare les éléments du tableau deux à deux et on les échange s'ils ne sont pas dans le bon ordre (ordre croissant). L'algorithme peut nécessiter plusieurs passes sur le tableau avant qu'il soit complètement trié. Ensuite, vous l'appliquerez au tableau suivant : | 5 | 3 | 2 | 1 |

Correction : *Algorithme Tri (t : tableau des entiers)*
Variables echange, n, temp : entiers
Début :
echange ← 1
n ← taille(t)
Tant que echange = 1
    echange← 0
    Pour i de 0 à n-2
        Si tab[i] > tab[i + 1]
            echange ← 1
            temp ← tab[i]
            tab[i] ← tab[i+1]
            tab[i+1] ← temp
            Fin Si
        Fin Pour
    Fin Tant que
Fin

5321→ 3521 → 3251 → 3215 → 2315 → 2135 → 2135 → 1235 → 1235 → 1235

---

2 Fusion de tableaux triés

Écrivez un programme permettant de fusionner deux tableaux d'entiers triés. L'algorithme devra générer aléatoirement les deux tableaux (taille et contenu), puis il utilisera le tri à bulle pour les trier et enfin, il les fusionnera en faisant attention à ce que le nouveau tableau soit lui aussi trié. Vous décomposerez l'algorithme en fonctions élémentaires (génération des tableaux, tri puis fusion).

Correction : *Algorithm Fusion*
Variables t1,t2,t_{new} : tableau d'entiers, i,j,t,w,n1,n2 : entiers
Début :
    t1 ← getTab(), t2 ← getTab()
    n1 ← taille(t1), n2 ← taille(t2)

```
    j ← 1, t ← 1
    Pour i de 1 à n1
        Si t2[j] < t1[i]
            t_new[t] ← t2[j]
            j ← j + 1
            i ← i − 1
        Sinon
            t_new[t] ← t1[i]
        Fin Si
        t← t+1
    Fin Pour
    Pour w de j à n2
        t_new[t] ← t2[w]
        t← t+1
    Fin Pour
Fin
```

---

3 Recherche rapide dans un tableau trié

Écrivez un programme permettant de rechercher rapidement une valeur dans un tableau. Le programme devra générer aléatoirement le tableau (taille et contenu), puis il utilisera le tri à bulle pour le trier et enfin, il recherchera la valeur demandée par l'utilisateur dans le tableau. La recherche devra être effectuée par dichotomie en exploitant le fait que les valeurs soient triées. Vous décomposerez l'algorithme en fonctions élémentaires (génération des tableaux, tri puis recherche)

Correction :

```
Algorithme Recherche
Variable t : tableau d'entiers, x : entier, b : boolean
Début
    t ← getTab()
    triBulle(t)
    Lire(x)
    b ← rechercheElement(t, x)
    Si (b = VRAI)
        Ecrire(t, "contient", x)
    Sinon
        Ecrire(t, "ne contient pas", x)
    Fin si
Fin

Fonction getTab()
Variable n, i : entier, t : tableau d'entiers
Début
    Ecrire("Donner la taille du tableau")
    Lire(n)
    Pour i ← 1 à n faire
        Ecrire("Entrer une valeur")
        Lire(t[i])
    Fin pour
    Retourner t
Fin

Fonction rechercheElement(tab : tableau des entiers, x : entier)
Variable i,j : entier
Début
    i ← 0
```

```
    j ← taille(tab) − 1
    Tantque (i ≤ j) faire
        Si (tab[(j + i)/2] = x)
            Retourner VRAI
        Sinon
            Si (tab[(j + i)/2] > x)
                j ← (j + i)/2 − 1
            Sinon
                i ← (j + i)/2 + 1
            Fin si
        Fin si
    Fin tantque
    Retourner FAUX
Fin
```

---

4 Multiplication matricielle

Écrivez un programme qui effectue la multiplication de deux matrices A et B. Le résultat de la multiplication sera mémorisé dans une troisième matrice C qui sera ensuite affichée. Pour mémoire, en multipliant une matrice A de dimensions n et m avec une matrice B de dimensions m et p on obtient une matrice C de dimensions n et p :

$$A(n, m) \times B(m, p) = C(n, p)$$

La multiplication de deux matrices se fait en multipliant les composantes des deux matrices lignes par colonnes :

$$c_{i,j} = \sum_{k=1}^{k=m} (a_{i,j} \times b_{k,j})$$

Vous décomposerez l'algorithme en fonctions élémentaires (création des matrices, affichage des matrices, produit matriciel).

Correction :

```
Procedure produitMatrices(A[n,m] :tableau d'entiers, B[m,p] :tableau d'entiers, C[n,p] :tableau d'entiers)
Variables i, j, k : entier
Debut
    Pour i ← 1 à n faire
        Pour j ← 1 à p faire
            C[i][j] ← 0
            Pour k ← 1 à m faire
                C[i][j] ← C[i][j] + A[i][k] * B[k][j]
            Fin pour
        Fin pour
    Fin pour
    Afficher(C)
Fin

Procedure LirerMatrice(A[n,m] :tableau d'entiers)
Variables
Debut
    Pour i ← 1 à n faire
        Pour j ← 1 à m faire
            Lire(A[i][j] )
        Fin pour
    Fin pour
Fin
```

```
Procedure AfficherMatrice(A[n,m] :tableau d'entiers)
Variables
Debut
    Pour i ← 1 à n faire
        Pour j ← 1 à m faire
            Ecrire(A[i][j] )
        Fin pour
    ALaLigne
    Fin pour
Fin
```

---

5 Triangle de Pascal

Écrivez un algorithme GenererPascal retournant un tableau 2D de hauteur n avec les coefficients du triangle de Pascal. Exemple pour n = 5 :

1				
1	1			
1	2	1		
1	3	3	1	
1	4	6	4	1

Correction :

```
Fonction GenererPascal(n : entier)
Variable t : tableau 2D d'entiers à taille n par n
Début
    Pour i ← 2 à n faire
        Pour j ← 1 à i faire
            Si (j = 1) ou (j > i − 1)
                t[i] [j] ← 1
            Sinon
                t[i] [j] ← t[i − 1] [j − 1] + t[i − 1] [j]
            Fin si
        Fin pour
    Fin pour
Fin
```
