

Correction TD2

MODULE ALGORITHMIQUE II

Chapitre 2 : Tableaux

Exercice 1 : lecture et écriture

Algorithme Exercice_1

variables

tableau note (12): entier

i: entier

Début

Pour i<--0 à 11

Ecrire ("la note du module", i+1, "est : ")

Lire (note [i])

FinPour

pour j<--0 à 11

Ecrire (note [i])

FinPour

Fin

Exercice 2 : inverser les éléments d'un tableau

Algorithme Exercice_2

variables

Tableau jeu(): entier

i,n : entier

Début

ECRIRE ("saisir une valeur: ")

Lire (n)

Redim jeu(n)

Pour i<--0 à n-1

ECRIRE ("donner le nombre n°", i+1)

LIRE (jeu(i))

FinPour

Pour i<--n-1 à 0 pas -1

Ecrire (jeu(i))

FinPour

Fin

Exercice 3

Algorithme exercice_3

variables

Tableau ListeElementsDistincts [], ListeElements[],ListeOccurences[]: entier;

NombreElementsDistincts, PositionElementDistinct, occurrence: entier

temp: booléen

temp ←vrai

NombreElementsDistincts<-- 0

Début

ECRIRE(" Entrez le nombre d'éléments dans le tableau: ")

LIRE (NombreElements)

Redim (ListeElements (NombreElements))

ECRIRE(" Entrez les éléments du tableau: ");

Pour i <--0 à NombreElements -1

LIRE (ListeElements[i])

//=====

// Pour chaque élément de la liste

// chercher le dans la liste des éléments distincts

// Si trouvé, incrémenter son nombre d'occurrences

// Si non trouvé, ajouter cet élément dans la liste des éléments distincts et ajouter '1' au tableau des occurrences

//=====

Pour i ← 0 à NombreElements-1

temp ← vrai;

pour j ← 0 à NombreElementsDistincts

Si (ListeElements[i]==ListeElementsDistincts[j]) alors

PositionElementDistinct = j

temp ← faux;

FinSi

FinPour

Si (temp) alors

ListeElementsDistincts[NombreElementsDistincts] = ListeElements[i];

ListeOccurrences[NombreElementsDistincts] = 1;

NombreElementsDistincts += 1;

Sinon

ListeOccurrences[PositionElementDistinct] += 1

FinSi

FinPour

pour k ← 0 à NombreElementsDistincts-1 faire

Si ListeOccurrences[k] > 1

occurrence ← occurrence+1

FinSi

Cette boucle cherche pour chaque élément de la liste si il est identique à un élément de la liste des éléments distincts afin de retenir sa position.

le changement de l'état de la variable booléen nous signale qu'il y a un élément qui se répète

Cette condition permet de remplir le 2ème tableau à chaque fois qu'on aura un élément différent des éléments du tableau des éléments distincts et ce grâce au booléen **temp**

Ecrire ("nombre total d'éléments en double est:", occurrence)

Fin

Simulation

ListeElements()

10	20	30	10	60	10	20	80	30	80
----	----	----	----	----	----	----	----	----	----

i ↑

ListeElementsDistincts()

10	20	30	60	80
----	----	----	----	----

j ↑

ListeOccurences()

3	2	2	1	2
---	---	---	---	---

Exercice 4

Algorithme exercice_4

variables

Tableau ListeElementsDistincts [], ListeElements[], ListeOccurences[]: entier;

NombreElementsDistincts, PositionElementDistinct, occurrence: entier

temp: booléen

temp ← vrai

NombreElementsDistincts ← 0

Début

ECRIRE(" Entrez le nombre d'éléments dans le tableau: ")

LIRE (NombreElements)

Redim (ListeElements (NombreElements))

ECRIRE(" Entrez les éléments du tableau: ");

Pour i ← 0 à NombreElements -1

LIRE (ListeElements[i])

```
//=====

// Pour chaque élément de la liste

//  chercher le dans la liste des éléments distincts

//  Si trouvé, incrémenter son nombre d'occurrences

//  Si non trouvé, ajouter cet élément dans la liste des éléments distincts et ajouter '1' au
tableau des occurrences

//=====

Pour i ← 0 à NombreElements-1

    temp ← vrai;

    pour j ← 0 à NombreElementsDistincts

        Si (ListeElements[i] == ListeElementsDistincts[j]) alors

            PositionElementDistinct = j

            temp ← faux;

        FinSi

    FinPour

    Si (temp) alors

        ListeElementsDistincts[NombreElementsDistincts] = ListeElements[i];

        ListeOccurrences[NombreElementsDistincts] = 1;

        NombreElementsDistincts += 1;

    Sinon

        ListeOccurrences[PositionElementDistinct] += 1

    FinSi

FinPour

Pour k ← 0 à NombreElementsDistincts

    ecrire (ListeElementsDistincts[k], " fois la température était de, ListeOccurrences[k], "°C")

FinPour
```

```
pour i ← 1 à NombreElementsDistincts
max = ListeOccurences[0]
Si (ListeOccurences[i] > max
    max = ListeOccurences[i]
    ecrire ("le nombre max de tableau est:", max )
FinSi
FinPour
Fin
```

Exercice 5

Algorithme exercice 5_triCroissant

Tableau Tab (): entier

i, j, n: entier

Ecrire ("saisir la taille du tableau")

Lire(n)

Redim(Tab(n))

Début

pour i ← 0 à n-1 faire

posmin = i

pour j ← i+1 à n faire

Si Tab[j] < Tab[posmin] Alors

Tab[j] = Tab[posmin]

posmin = j

FinSi

FinPour

A=Tab[i]

Tab[i]=Tab[posmin]

Tab[posmin]=A

FinPour

Fin

Algorithme exercice 5_ordreDécroissant

Tableau Tab(): entier

i,n: entier

inversion: booléen

inversion=True

Ecrire ("saisir la taille du tableau")

Lire(n)

Redim(Tab(n))

Debut

TantQue (inversion)

Pour i<--0 n-1

inversion=False

Si (Tab[i]>Tab[i+1]) alors

Inversion ← True

A=Tab[i]

Tab[i]=Tab[i+1]

Tab[i+1]=A

FinSi

FinPour

FinTantQue

Fin

Algorithme exercice 5_AjoutZéro

Tableau Tab(): entier

i,n: entier

inversion: booléen

inversion \leftarrow *True*

Ecrire ("saisir la taille du tableau")

Lire(n)

Redim(Tab(n))

Debut

pour i \leftarrow *-1* à *NombreElements-1*

temp \leftarrow *vrai*;

Si(*ListeElements[i]*==0)

Temp \leftarrow *faux*

FinSi

Si (*temp*==*vrai*)

ListeElementsSansZero[*NombreSansZero*] = *ListeElements[i]*;

NombreSansZero += 1;

FinSi

FinPour

pour j \leftarrow *NombreSansZero* à *NombreElements-1*

ListeElementsSansZero[j]=0

FinPour

Fin

Exercice 6

Algorithme exercice 6_insertion_tabeleauTrié

variables

Tableau T (): entier

N, p, v, : entier

Début

Lire(v);

M←-N+1

pour i←-0 à i←-N-1

si T[i]>v

pos=i

finpour

pour k←-M-1 à pos pas -1 faire

T[k] ← T[k-1]

fpour

T[pos] ← v

Fin

Algorithme exercice 6_insertion_finTableau

variables

tableau Tab(), Ntab(): entier

i,n: entier

Ecrire ("saisir la taille du tableau")

Lire(NombreElement)

Redim(Tab(NombreElement))

M=NombreElements+1

Debut

pour i<--0 à i <--NombreElements-1

Ntab[i] = Tab[i];

Finpour

Ntab[M-1]=p

Fin

Algorithme Suppression

variables

tableau Tab (): entier

N, p, v : entier

Début

Ecrire ("saisir la taille du tableau")

Lire(N)

Redim(Tab(N))

Ecrire ("saisir la position à supprimer")

Lire (p);

pour i<--0 à p-1

Ntab[i] = Tab[i];

Finpour

pour k= p+1 à N – 1 faire

Ntab[k-1] = Tab[k]

finPour

Fin

Exercice 8

Algorithme exercice_8

variables

tableau Tab () : entier

N, p, v : entier

temp : booléen

temp \leftarrow vrai

Début

Ecrire ("saisir la taille du tableau")

Lire (n)

Redim(Tab(N))

pour i \leftarrow 0 à i \leftarrow n-2

si Tab[i+1]=T[i]+1

 Ecrire (Tab[i], Tab[i+1], " sont des éléments consécutives")

 temp \leftarrow false

FinSi

Finpour

si (temp)

 Ecrire (pas d'éléments consécutives")

Finsi

Fin

Exercice 7 :

Algorithme exercice_7

Variables

Tableau A(3)(3), B(3)(3), C(3)(3) : entier

$i, j : \text{entier}$

Début

Pour $i \leftarrow 0$ à 2

Pour $j \leftarrow 0$ à 2

Lire $(A(i)(j))$

Ecrire $((A(i)(j)))$

finPour

finPour

Pour $i \leftarrow 0$ à 2

Pour $j \leftarrow 0$ à 2

Lire $(B(i)(j))$

Ecrire $((B(i)(j)))$

finPour

finPour

Pour $i \leftarrow 0$ à 2

Pour $j \leftarrow 0$ à 2

$C(i)(j) = A(i)(j) - B(i)(j)$

finPour

finPour

Pour $i \leftarrow 0$ à 2

Pour $j \leftarrow 0$ à 2

Ecrire $(C(i)(j))$

finPour

finPour

fin

Algorithme multiplicationMatrice

Variables

Tableau $A()()$, $B()()$, $C()()$: entier

i, j : entier

Début

Lire (n)

Lire (m)

Lire (p)

Pour $i \leftarrow 0$ à $n-1$

pour $j \leftarrow 0$ à $p-1$

$C(i)(j) = 0$;

pour $k \leftarrow 0$ à $m-1$

$C(i)(j) = C(i)(j) + A(i)(k) * B(k)(j)$

FinPour

FinPour

Lire("Matrice résultat C :")

pour $i \leftarrow 0$ à $n-1$

pour $j \leftarrow 0$ à $p-1$

Lire ($C(i)(j)$)

finPour

finPour

Algorithme transposé

Variables

Tableau $A()()$, $B()()$, $C()()$: entier

i, j : entier

Début

Lire (n)

Lire (m)

Pour i ← 0 à n-1

pour J ← 0 à m-1

Lire(A(i)(j))

FinPourj

FinPouri

/ Affectation de la matrice transposée à B */*

Pour i ← 0 à n-1

pour J ← 0 à m-1

B(j)(i)=A(i)(j)

FinPourj

FinPouri

/ Edition du résultat */*

/ Attention: maintenant le rôle de N et M est inversé. */*

Pour i ← 0 à m-1

pour J ← 0 à n-1

Lire(A(i)(j))

FinPourj

FinPouri

Fin