



Afpa

se former, avancer

L'ALGORITHME

Corrections exercices cours



Corrections

Partie 2 du cours



Exercice 4.1

```
// Programme MoyenneNotes
// Auteur : Jérôme BOEBION
// Description : Calcul de la moyenne des notes
Programme MoyenneNotes

// déclaration des variables
Variables
    compteur : entier
    résultat : réel
    notes tableau[9] : réel

Début
    // initialisation
    resultat <- 0
    Pour compteur Allant de 1 à 9 Faire
        Ecrire("Entrez une note ?")
        lire(notes[compteur])
        resultat <- resultat + notes[compteur]
        compteur <- compteur + 1
    FinPour
    Ecrire("la moyenne est " + resultat/9)
Fin
```

```
VARIABLES
compteur EST_DU_TYPE NOMBRE
resultat EST_DU_TYPE NOMBRE
moyenne EST_DU_TYPE NOMBRE
notes EST_DU_TYPE LISTE

DEBUT_ALGORITHME
// initialisation
resultat PREND_LA_VALEUR 0
POUR compteur ALLANT_DE 1 A 9
    DEBUT_POUR
        AFFICHER "Entrez une note ?"
        LIRE notes[compteur]
        resultat PREND_LA_VALEUR resultat + notes[compteur]
    FIN_POUR
    moyenne PREND_LA_VALEUR resultat / 9
    AFFICHER "La moyenne est "
    AFFICHER moyenne

FIN_ALGORITHME
```



Exercice 4.2

```
procédure tri_insertion(tableau)
  pour i de 1 à longueur(tableau) - 1 faire
    clé = tableau[i]
    j = i - 1
    tant que j >= 0 et tableau[j] > clé faire
      tableau[j + 1] = tableau[j]
      j = j - 1
    fin tant que
    tableau[j + 1] = clé
  fin pour
fin procédure
```

```
FONCTIONS_UTILISEES
FONCTION afficherTableau(tab)
  VARIABLES_FONCTION
  i EST_DU_TYPE NOMBRE
  DEBUT_FONCTION
  POUR i ALLANT_DE 0 A tableau.length-1
    DEBUT_POUR
    AFFICHER tableau[i]
    FIN_POUR
  AFFICHER " "
  FIN_FONCTION
FONCTION initialisationTableau( )
  VARIABLES_FONCTION
  DEBUT_FONCTION
  // initialisation du tableau avec des valeurs
  tableau[0] PREND_LA_VALEUR 4:2:5:1:3
  FIN_FONCTION
VARIABLES
tableau EST_DU_TYPE LISTE
indice EST_DU_TYPE NOMBRE
cle EST_DU_TYPE NOMBRE
j EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
  // initialisation du tableau avec des valeurs
  APPELER_FONCTION initialisationTableau()
  // affichage du tableau initial
  AFFICHER "Tableau non trié"
  APPELER_FONCTION afficherTableau(tableau)
  // trie du tableau
  POUR indice ALLANT_DE 1 A tableau.length-1
    DEBUT_POUR
    cle PREND_LA_VALEUR tableau[indice]
    j PREND_LA_VALEUR indice-1
    TANT_QUE (j >= 0 ET tableau[j] > cle) FAIRE
      DEBUT_TANT_QUE
      tableau[j+1] PREND_LA_VALEUR tableau[j]
      j PREND_LA_VALEUR j - 1
      FIN_TANT_QUE
    tableau[j + 1] PREND_LA_VALEUR cle
    FIN_POUR
  APPELER_FONCTION afficherTableau(tableau)
FIN_ALGORITHME
```

Exercice 4.3

Notes :

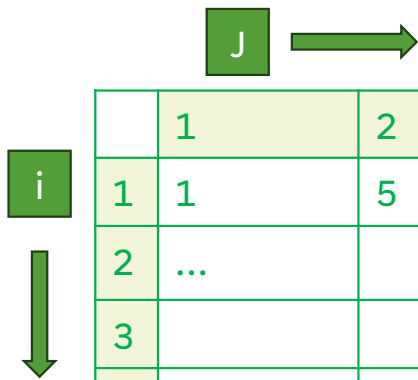
- n correspond à la taille du Dico
- max et min sont définis à partir de la taille du dico
- Indice indique le milieu de chaque sous-ensemble de l'élément à comparer.

Si le mot se situe avant le point de comparaison, alors la borne supérieure change, la borne inférieure ne bouge pas

Sinon c'est l'inverse...

```
1  FONCTIONS_UTILISEES
2  VARIABLES
3      max EST_DU_TYPE NOMBRE
4      min EST_DU_TYPE NOMBRE
5      indice EST_DU_TYPE NOMBRE
6      valeurRecherchee EST_DU_TYPE NOMBRE
7      dico EST_DU_TYPE LISTE
8      temp EST_DU_TYPE NOMBRE
9      flag EST_DU_TYPE NOMBRE
10 DEBUT_ALGORITHME
11     //initialisation des variables
12     dico[0] PREND_LA_VALEUR 1:2:3:4:5:6:7:8:9:10
13     flag PREND_LA_VALEUR 0
14     AFFICHER "nombre à verifier ?"
15     LIRE valeurRecherchee
16     max PREND_LA_VALEUR dico.length-1
17     min PREND_LA_VALEUR 0
18     TANT_QUE (min <= max ET flag == 0) FAIRE
19         DEBUT_TANT_QUE
20             //determine le milieu
21             indice PREND_LA_VALEUR floor((max+min)/2)
22             //Si la valeur est sur l'indice du milieu c'est trouvé
23             SI (dico[indice] == valeurRecherchee) ALORS
24                 DEBUT_SI
25                     AFFICHER "La valeur existe"
26                     flag PREND_LA_VALEUR 1
27                 FIN_SI
28             SINON
29                 DEBUT_SINON
30                     //Sinon on modifie min ou max selon si la valeur se trouve dans le sous ensemble
31                     SI (dico[indice] < valeurRecherchee) ALORS
32                         DEBUT_SI
33                             min PREND_LA_VALEUR indice+1
34                         FIN_SI
35                     SINON
36                         DEBUT_SINON
37                             max PREND_LA_VALEUR indice-1
38                         FIN_SINON
39                     FIN_SINON
40                 FIN_TANT_QUE
41             //dans le cas où flag n'a pas bougé c'est que valeur n'existe pas
42             SI (flag == 0) ALORS
43                 DEBUT_SI
44                     AFFICHER "la valeur n'existe pas"
45                 FIN_SI
46     FIN_ALGORITHME
```

Exercice 4.4



| | 1 | 2 | 3 | 4 | 5 |
|---|-----|---|---|---|---|
| 1 | 1 | 5 | 3 | 4 | 2 |
| 2 | ... | | | | |
| 3 | | | | | |
| 4 | | | | | |
| 5 | | | | | |

Notes :

Le principe de la recherche dans un tableau à deux dimensions est strictement le même que pour un tableau à une dimension.

La seule chose qui va changer, c'est qu'ici le balayage requiert deux boucles imbriquées.

```
// Programme RechercheMax
// Auteur : Jérôme BOEBION
// Description : recherche d'une valeur Max dans un tableau
Programme RechercheMax
Variables
    i, iMax, j, jMax : entier
    T[12, 8] : entier
Début
    iMax <- 1
    jMax <- 1
    Pour i Allant de 1 à 12 Faire
        Pour j Allant de 1 à 8 Faire
            // si l'élément est supérieur
            // on conserve alors les indices
            Si T[i,j] > T[iMax, jMax] Alors
                iMax := i
                jMax := j
            FinSi
            // on incremente la colonne
            j := j + 1
        FinPour
        // on incremente la ligne
        i := i + 1
    FinPour
    Ecrire("le plus grand élément est ", T[iMax, jMax])
    Ecrire("Il se trouve aux indices ", iMax, "/", jMax)
Fin
```

Exercice 5.1

```
// Programme CompterX
// Auteur : Jérôme BOEBION
// Description : compte le nombre d'une valeur dans un tableau
Programme CompterX
Variables
    occurrences : Entier
    x, indice : Entier
    tab : tableau de [n] Entier

Début
    Ecrire("Nombre à rechercher entre 0-100 ?")
    Lire(x)

    Pour indice allant de 1 à n faire
        occurrences <- 0
        Si tab[indice] = x alors
            occurrences <- occurrences + 1
        FinSi
        indice <- indice + 1
    FinPour
    Ecrire(x, " : ", occurrences, " occurrence(s)")
Fin
```



Exercice 5.2

```
// Programme Recherche_Occurences
// Auteur : Jérôme BOEBION
// Description : compte le nombre de chaque valeur dans un tableau
Programme Recherche_Occurences
// Déclaration
Variables
    tabOccurences : tableau de [n] Entier
    tabValeurs : tableau de [n] Entier
    valeur : Entier

Début
    // boucle d'initialisation à zéro du tableau des occurrences
    Pour i allant de 1 à n Faire
        tabOccurences[i] <- 0
    FinPour
    // ... boucle de détection des occurrences
    Pour i allant de 1 à n Faire
        // initialisation du nombre d'occurrences
        occurrences <- 0
        valeur <- tabValeurs[i] // sauvegarde de la valeur de l'indice pour l'enregistrement
        Pour j allant de 1 à n Alors
            Si valeur = tabValeurs[j] Alors // si valeur est présente alors je compte
                occurrences <- occurrences + 1
            FinSi
            j <- j + 1
        FinPour
        // j'enregistre à l'indice correspond à valeur le nombre d'occurrences
        tabOccurences[valeur] <- occurrences
        i <- i + 1
    Fin Pour
    // ... boucle d'affichage des occurrences ...
Fin
```




```

// Programme Hanoï
// Auteur : Jérôme BOEBION
// Description : Jeu de la tour d'Hanoï
Programme Hanoï
Fonction déplacer(nbDisques,départ,arrivée,intermédiaire)
Déclarations
    nbDisques, départ, arrivée, intermédiaire : entier
Début
    Si nbDisques = 1 alors
        écrire(départ, " -> ", arrivée)
    Sinon
        déplacer(nbDisques-1, départ, intermédiaire, arrivée)
        déplacer(1, départ, arrivée, intermédiaire)
        déplacer(nbDisques-1, intermédiaire, arrivée, départ)
    Finsi
Fin

// Déclaration
Variable
    nbDisques : entier

Début
    écrire('Entrez le nombre de disques :')
    lire(nbDisques)
    // appel initial
    déplacer(nbDisques, 1, 2, 3)
Fin

```

Exercice 6

La procédure récursive de déplacement sera :

Déplacer(nbDisques, départ, arrivée, intermédiaire)

- nbDisques : nombre de disques à déplacer
- Départ : numéro de la tour de départ
- Arrivée : numéro de la tour cible
- Intermédiaire : numéro de la tour restante
- Exemple du 1er déplacement du schéma précédent :
 - déplacer(1,1,3,2)





Formateur

Jérôme BOEBION

Concepteur Développeur d'Applications

