UNIVERSITE PARIS DAUPHINE

IUP Génie Mathématique et Informatique

IUP Génie Mathématique et Informatique - 2ème année
Mise à Niveau Informatique
2003 - 2004

Algorithmique - Exercices corrigés

Maude Manouvrier

1 Exercices simples

1.1 Enoncé

Faire les exercices suivants, individuellement et sur feuille :

1. Écrire un algorithme permettant la saisie d'une note et son affichage. Afficher en plus un message si la note est en dessous de la moyenne.
2. Ajouter à l'algorithme précédent une vérification de la note (nombre compris entre 0 et 20). Une note incorrecte ne doit pas être affichée (message d'erreur).
3. Ecrire un algorithme permettant de saisir N notes (N connu et fixé). *On utilisera un compteur et les mot clé POUR et FIN POUR.*
4. Même exercice que précédemment, mais en comptant les notes incorrectes et les notes en dessus de la moyenne. Afficher les résultats.
5. Écrire un algorithme permettant la saisie de N notes (N connu et fixé) et de calculer leur moyenne.
6. Même exercice que précédemment, mais en ne prenant pas en compte les notes incorrectes.
7. Alléger le traitement de la moyenne en imposant que les notes soient correctes (re-saisir la note jusqu'à ce qu'elle soit correcte).
8. Même exercice que précédemment, mais en ajoutant le calcul de la variance.
   Rappel : la variance est égale à $\sigma^2 = \frac{\sum_{i=1}^{N}(X_i-\mu)^2}{N}$, avec X_i la $i$ème note et μ la moyenne.
9. Écrire un algorithme permettant de calculer et d'afficher la valeur de x^y, les variables x et y étant des entiers positifs saisis par l'utilisateur (sans utiliser de fonction pré-définie puissance).
10. Écrire un algorithme qui permette de calculer x^y lorsque y est un entier positif ou négatif en reprenant l'algorithme précédent (que l'on appellera fonction puissance).
11. Écrire un algorithme permettant de résoudre l'équation $ax^2 + bx + c = 0$.
12. Écrire un algorithme permettant, à partir de deux tableaux d'entiers T_1 et T_2 contenant chacun N entiers non triés, de modifier les tableaux de telle sorte que T_1 contienne tous les entiers pairs contenus à l'origine dans T_1 et T_2 et que T_2 contienne tous les entiers impairs contenus à l'origine dans T_1 et T_2. On a pour hypothèse de départ que T_1 et T_2 réunis contiennent N entiers pairs et N entiers impairs.
    Par exemple, si au départ T_1 = [0,7,5,3] et T_2 = [4,2,9,6],
    alors à la fin de l'algorithme, T_1 = [0,4,2,6] et T_2 = [7,5,9,3].

NB: Lorsqu'un exercice reprend l'algorithme d'un exercice précédent, vous pouvez uniquement indiquer ce qui change et l'emplacement de ce changement.

1.2 Corrigé

1.2.1 Exercice 1

```
ALGORITHME de saisie et d'affichage d'une note
           avec avertissement quand la note est sous la moyenne

VARIABLES : note : entier

Début

Afficher("Saisissez une note :")
Lire(note)

SI (note >= 0) ET (note < 10)
  ALORS Afficher("La note est en dessous de la moyenne");
FIN SI

Fin
```

1.2.2 Exercice 2

```
ALGORITHME de saisie et d'affichage d'une note
           avec avertissement quand la note est sous la moyenne
           et vérification de la cohérence de la note

VARIABLES :
   note : entier

Début

Afficher("Saisissez une note :")
Lire(note)

SI (note >= 0) ET (note <= 20)
  ALORS SI (note < 10)
         ALORS Afficher("La note est en dessous de la moyenne");
        FIN SI
SINON Afficher("Note incorrecte");
FIN SI

Fin
```

1.2.3 Exercice 3

```
ALGORITHME de saisie de N notes - N connu et donné

VARIABLES :
   note[] : tableaux de N entiers indicés de 0 à N
   compteur : entier

Début

POUR compteur ALLANT DE 1 à N PAR PAS DE 1
  Afficher("Saisissez une note :")
  /* On place la note à la position (compteur -1) du tableau */
  Lire(note[compteur-1])
FIN POUR

Fin
```

1.2.4 Exercice 4

```
ALGORITHME de saisie de N notes - N connu et donné
           avec décompte des notes incorrectes et des notes sous la moyenne

VARIABLES :
   note[] : tableaux de N entiers indicés de 0 à N
   compteur : entier
   nb_incorrectes : entier /* nombre de notes incorrectes */
   nb_ssmoyenne : entier   /* nombre de notes sous la moyenne */

Début

nb_incorrectes = 0
nb_ssmoyenne = 0

POUR compteur ALLANT DE 1 à N PAR PAS DE 1

  Afficher("Saisissez une note :")
  /* On place la note à la position (compteur -1) du tableau */
  Lire(note[compteur-1])

  SI (note >= 0) ET (note <= 20)
     ALORS SI (note < 10)
              ALORS nb_ssmoyenne =  nb_ssmoyenne +1
           FIN SI
  SINON nb_incorrectes = nb_incorrectes +1
  FIN SI

FIN POUR
```

```
Afficher("Nombre de notes incorrectes : ", nb_incorrectes)
Afficher("Nombre de notes sous la moyenne : ", nb_ssmoyenne)

Fin
```

1.2.5 Exercice 5

```
ALGORITHME de saisie de N notes - N connu et donné
           et calcul de la moyenne

VARIABLES :
   note[] : tableaux de N entiers indicés de 0 à N
   compteur : entier
   somme : entier

Début

somme = 0

POUR compteur ALLANT DE 1 à N PAR PAS DE 1

  Afficher("Saisissez une note :")
  /* On place la note à la position (compteur -1) du tableau */
  Lire(note[compteur-1])

  somme = somme + note

FIN POUR

Afficher("La moyenne des ", N, "notes saisies est : ", (somme/N))
Fin
```

1.2.6 Exercice 6

```
ALGORITHME de saisie de N notes - N connu et donné
           et calcul de la moyenne sans prendre en compte
           les notes incorrectes

VARIABLES :
   note[] : tableaux de N entiers indicés de 0 à N
   compteur : entier
   somme : entier
   /* nombre de notes correctes (comprises entre 0 et 20) */
   nb_correctes : entier


Début

somme = 0
nb_correctes = 0

POUR compteur ALLANT DE 1 à N PAR PAS DE 1

  Afficher("Saisissez une note :")
  /* On place la note à la position (compteur -1) du tableau */
  Lire(note[compteur-1])

  SI (note >= 0) ET (note <= 20)
      ALORS
         somme = somme + note
         nb_correctes = nb_correctes +1
  FIN SI

FIN POUR

SI (nb_correctes>0)
  ALORS Afficher("La moyenne des ", N,
                 "notes saisies est : ", (somme/nb_correctes))
SINON Afficher("Aucune note correcte donc pas de moyenne")

Fin
```

1.2.7 Exercice 7

```
ALGORITHME de saisie de N notes - N connu et donné
           et calcul de la moyenne en faisant resaisir les notes incorrectes

VARIABLES :
   note[] : tableaux de N entiers indicés de 0 à N
   compteur : entier
   somme : entier

Début
somme = 0
POUR compteur ALLANT DE 1 à N PAR PAS DE 1

  Afficher("Saisissez une note :")
  /* On place la note à la position (compteur -1) du tableau */
  Lire(note[compteur-1])

  TANT QUE (note < 0) OU (note > 20) FAIRE
     Afficher("Note incorrecte - Saisissez à nouveau une note :")
     /* On place la note à la position (compteur -1) du tableau */
     Lire(note[compteur-1])
  FIN TANT QUE

  somme = somme + note
FIN POUR

Afficher("La moyenne des ", N, "notes saisies est : ", (somme/N))

Fin
```

1.2.8 Exercice 8

```
ALGORITHME de saisie de N notes - N connu et donné
           et calcul de la moyenne et de la variance
           en faisant resaisir les notes incorrectes


VARIABLES : note[] : tableaux de N entiers indicés de 0 à N
            compteur : entier
            somme : entier
            moyenne : entier
            sigma_2 : entier

Début
somme = 0
POUR compteur ALLANT DE 1 à N PAR PAS DE 1

  Afficher("Saisissez une note :")
  /* On place la note à la position (compteur -1) du tableau */
  Lire(note[compteur-1])

  TANT QUE (note < 0) OU (note > 20) FAIRE
     Afficher("Note incorrecte - Saisissez à nouveau une note :")
     /* On place la note à la position (compteur -1) du tableau */
     Lire(note[compteur-1])
  FIN TANT QUE

  somme = somme + note

FIN POUR

moyenne = somme / N

/* On remet somme à 0 pour calculer la variance */
somme = 0

/* Calcul de la variance */
POUR compteur ALLANT DE 1 à N PAR PAS DE 1
   somme = somme + (note[compteur-1] - moyenne)^2
FIN POUR

sigma_2 = somme / N

Afficher("La moyenne des ", N, "notes saisies est : ", moyenne)
Afficher("La variance est :", sigma_2)

Fin
```

1.2.9 Exercice 9

```
ALGORITHME de RÉSOLUTION DE X PUISSANCE Y, Y entier positif

VARIABLE
   x : entier saisi par l'utilisateur
   y : puissance saisie par l'utilisateur
   compteur : compteur pour la boucle
   resultat : valeur résultat de x puissance y

Début

Afficher("Saisie de x")
Lire(x)
Afficher("Saisie de y")
Lire(y)

SI y = 0
  ALORS resultat = 1
SINON
   SI y = 1
    ALORS resultat = x
   SINON
      resultat = x
      compteur = 0
      TANT QUE compteur < (y-1) FAIRE
        resultat = resultat * x;
        compteur = compteur + 1;
      FIN TANT QUE
   FIN SI
FIN SI

Afficher("La valeur de ", x, " puissance ", y, " est ", resultat)

Fin
```

1.2.10 Exercice 10

On appelle *puissance*, l'algorithme précédent où l'affichage final a été remplacé par `Retourner(resultat)`.

```
ALGORITHME de RÉSOLUTION DE X PUISSANCE Y

VARIABLE
   x : entier saisi par l'utilisateur
   y : puissance saisie par l'utilisateur
   compteur : compteur pour la boucle
   resultat : valeur résultat de x puissance y

Début

Afficher("Saisie de x")
Lire(x)
Afficher("Saisie de y")
Lire(y)

SI y < 0
  ALORS
    SI (x!=0)
     ALORS
       resultat = 1 / puissance(x,-y)
       Afficher("La valeur de ", x, " puissance ", y, " est ", resultat)
     SINON Afficher("Erreur'')
    FIN SI
SINON
   Afficher("La valeur de ", x, " puissance ", y, " est ", puissance(x,y))
FIN SI

Fin
```

1.2.11 Exercice 11

```
ALGORITHME de résolution de l'équation ax^2+bx+c=0

VARIABLE
    a : entier
    b : entier
    c : entier
    Delta : réel
    Racine_Delta:entier

Début

Afficher("Saisie du coefficient a")
Lire(a)
Afficher("Saisie du coefficient b")
Lire(b)
Afficher("Saisie du coefficient c")
Lire(c)

SI (a <> 0) ALORS
     Delta = b^2 -4*a*c
     SI Delta > 0 ALORS
       Racine_Delta = Racine(Delta)
       Afficher("Les solutions sont ", (-b + Racine_Delta) / (2*a))
       Afficher("et", (-b - Racine_Delta) / (2*a))
     SINON
       SI  Delta = 0 ALORS
         Afficher("Solution double :",-b/(2*a))
       SINON /* Delta < 0*/
         Calculer Racine_Delta = Racine(-Delta)
         Afficher(Les solutions sont ", (-b/(2*a)), "+ i",
                   (Racine_Delta) / (2*a))
         Afficher("et" (-b/(2*a)), "- i", (Racine_Delta) / (2*a))
       FIN SI
     FIN SI
SINON
    SI b<>0
      ALORS Afficher("Résultat", -c/b)
    SINON
      SI c<>0
        ALORS Afficher("Pas de solution")
      SINON Afficher("Un infinité de solutions")
      FIN SI
    FIN SI
FIN SI
Fin
```

1.2.12 Exercice 12

A partir de deux tableaux d'entiers T_1 et T_2 contenant chacun N entiers non triés (les deux tableaux réunis contenant N entiers pairs et N entiers impairs), on souhaite écrire un algorithme permettant de modifier les tableaux T_1 et T_2 de telle sorte que T_1 contienne tous les entiers pairs contenus à l'origine dans T_1 et T_2 et que T_2 contienne tous les entiers impairs contenus à l'origine dans T_1 et T_2.
Cet algorithme peut s'écrire de plusieurs manières plus ou moins optimales.
On suppose les deux tableaux T_1 et T_2 sont déjà saisis par l'utilisateur et que les indices des tableaux vont de 0 à $(N-1)$.
On suppose qu'on a une fonction `Pair(i)` qui retourne `VRAI` sii est pair et `FAUX` sinon (en vérifiant que le reste de la division de `i` par 2 est zéro).

Premier algorithme

```
ALGORITHME TABLEAUX D'ENTIERS PAIRS ET IMPAIRS EN UTILISANT DEUX AUTRES TABLEAUX

VARIABLE
T1[] : premier tableau d'entiers de taille N
T2[] : deuxième tableau d'entiers de taille N
Pos1 : entier indiquant la position courante dans le tableau T1
Pos2 : entier indiquant la position courante dans le tableau T2
T3[] : tableau d'entiers de taille N vide devant contenir des entiers pairs
T4[] : tableau d'entiers de taille N vide devant contenir des entiers impairs

Début
 Pos1=0  Pos2=0
 TANT QUE (Pos1<N) FAIRE
   SI Pair(T1[Pos1])=VRAI ALORS FAIRE
       T3[Pos1]=T1[Pos1]
       Pos1=Pos1+1
   FIN SI
   SINON /* T1[Pos1] est impair */ FAIRE
     T4[Pos1]=T1[Pos1]
     Pos1=Pos1+1
   FIN SINON
 FIN TANT QUE

 TANT QUE (Pos2<N) FAIRE
   SI Pair(T2[Pos2])=VRAI ALORS FAIRE
      T3[Pos2]=T2[Pos2]
      Pos2=Pos2+1
   FIN SI
   SINON /* T2[Pos2] est impair */ FAIRE
     T4[Pos2]=T2[Pos2]
     Pos2=Pos2+1
   FIN SINON
 FIN TANT QUE
Fin
```

Deuxième algorithme

```
ALGORITHME TABLEAUX D'ENTIERS PAIRS ET IMPAIRS
SANS UTILISER DE TABLEAU SUPPLÉMENTAIRE
MAIS EN PARCOURANT CHAQUE TABLEAU AU MOINS UNE FOIS

VARIABLE
T1[] : premier tableau de N entiers
T2[] : deuxième tableau de N entiers
Pos1 : entier indiquant la position courante dans le tableau T1
Pos2 : entier indiquant la position courante dans le tableau T2
EntierEchangé: entier utilisé pour faire un échange

Début

 Pos1=0
 Pos2=0

 FAIRE

  /* On cherche la position du premier entier impair dans T1 */
  TANT QUE (Pair(T1[Pos1])=VRAI ET Pos1<N) FAIRE
     Pos1=Pos1+1
  FIN TANT QUE

  /* On cherche la position du premier entier pair dans T2 */
  TANT QUE (Pair(T2[Pos2])=FAUX ET Pos2<N) FAIRE
      Pos2=Pos2+1
  FIN TANT QUE

  /* On échange le premier entier non pair de T1 avec le premier */
  /* entier non impair de T2 */
  SI (Pos1<N) ET (Pos2<N) ALORS FAIRE
      EntierEchangé=T1[Pos1]
      T1[Pos1]=T2[Pos2]
      T2[Pos2]=EntierEchangé
      Pos1=Pos1+1
      Pos2=Pos2+1
  FIN SI

 TANT QUE (Pos1<N) ET (Pos2<N)
Fin
```

Troisième algorithme

```
ALGORITHME TABLEAUX D'ENTIERS PAIRS ET IMPAIRS
SANS UTILISER DE TABLEAU SUPPLÉMENTAIRE
ET EN NE PARCOURANT AU PIRE QU'UN TABLEAU SI LES DEUX TABLEAUX
CONTIENNENT DEJA AU DÉPART UNIQUEMENT DES ENTIERS PAIRS
OU UNIQUEMENT  DES ENTIER IMPAIRS

VARIABLE
T1[] : premier tableau de N entiers
T2[] : deuxième tableau de N entiers
Pos1 : entier indiquant la position courante dans le tableau T1
Pos2 : entier indiquant la position courante dans le tableau T2
Pos3 : entier indiquant la position du premier entier mal placé

EntierEchangé: entier utilisé pour faire un échange

Début

 Pos3=0

 POUR Pos1 ALLANT DE 0 à (N-1) PAR PAS DE 1 FAIRE

   /* si un entier impair est stocké dans T1 alors un entier pair  */
   /* est stocké dans T2 puisqu'il y a au total N entiers pairs et */
   /* N entiers impairs.                                           */
   /* On va donc parcourir au moins une fois T1 mais               */
   /* peut être jamais T2 si T1 ne contient que des entiers pairs  */

   SI Pair(T1[Pos1])=FAUX ALORS FAIRE

     /* Recherche de le premier entier mal placé (donc pair) dans T2 */
     TANT QUE (Pair(T2[Pos2])=FAUX ET Pos2 < N) FAIRE
       Pos2=Pos2+1
     FIN TANT QUE

     EntierEchangé=T1[Pos1]
     T1[Pos1]=T2[Pos2]
     T2[Pos2]=EntierEchangé
     Pos2=Pos2+1

   FIN SI

 FIN POUR

Fin
```

2 Recherche, tri et insertion dans un tableau ou une liste d'entiers

2.1 Recherche séquentielle

```
ALGORITHME de recherche séquentielle dans un tableau de N entiers

Variables :
   T : tableau de N entiers (donné)
   valeur_recherchee: entier (donnée)
   trouve : boolean
   compteur : entier

Début

/* parcours du tableau jusqu'à obtention de la valeur recherchée */
POUR compteur ALLANT DE 0 à (N-1) PAR PAS DE 1 FAIRE

  SI T[compteur] = valeur_recherchee ALORS trouve = true
  SINON trouve = false
  FIN SI

FIN POUR

Fin
```

2.2 Recherche dichotomique

```
ALGORITHME de recherche dichotomique dans un tableau de N entiers


Variables :
   T : tableau de N entiers (donné)
   valeur_recherchee: entier (donnée)
   premier : entier
   dernier : entier
   milieu : entier
   compteur : entier
   trouve : boolean

Début

/* Initialisation */
premier = 0
dernier = N-1
trouve = false

/* Tant qu'on a pas trouve l'élément ou qu'on n'a pas regardé  */
/* tous les sous-tableaux où peut se trouver la valeur recherchée :*/
TANT QUE premier <= dernier ET trouve = false FAIRE

   /* Calcul de la position de l'élement situé au milieu du sous-tableau */
   milieu = (premier + dernier) DIV 2

   SI T[milieu] = valeur_recherchee
     ALORS trouve = true
   /* Si ce n'est pas la valeur recherchée */
   SINON

      /* Si c'est une valeur plus grande que la valeur recherchée */
      /* on regarde le sous-tableau de gauche */
      SI T[milieu] > valeur_recherchee ALORS dernier = milieu -1

      /* Sinon on regarde le sous-tableau de droite */
      SINON premier = milieu +1
      FIN SI

   FIN SI

FIN TANT QUE
Fin
```

2.3 Tri par sélection ordinaire

```
ALGORITHME de tri par sélection ordinaire d'un tableau de N entiers

Variables :
  T : tableau de n entiers (indicés à partir de 0)
  min: entier
  compteur : entier
  temp : entier
  indice_ele_courant : entier

Début

/* On parcours le tableau du début à la fin */
POUR indice_ele_courant ALLANT DE 0 à (n-1), PAR PAS DE 1 FAIRE

   /*Affectation min avec l'élément courant de la liste*/
   min=T[indice_ele_courant]

   /* On parcours le tableau de la position du minimum à la fin */
   POUR compteur ALLANT DE indice_ele_courant+1 à (n-1), PAR PAS DE 1 FAIRE

       /* Si on a un élément plus petit que le minimum courant */
       SI min > T[compteur]
          ALORS
            /* échange */
            temp = T[indice_ele_courant]
            T[indice_ele_courant] = T[compteur]
            T[compteur] = temp

            min= T[indice_ele_courant]
       FIN SI

   FIN POUR

 FIN POUR

Fin
```

2.4 Tri à bulle

```
ALGORITHME de tri à bulle dans un tableau de n entiers

Variables :
  T: tableau de n entiers
  compteur1 : entier
  compteur2 : entier
  temp : entier


Début
/* Parcours du tableau du début à la fin */
POUR compteur1 ALLANT DE 0 à (n-1), PAR PAS DE 1 FAIRE

  /* Parcours du tableau de la fin à la position courante */
  POUR compteur2 ALLANT DE (n-1) à compteur1, PAR PAS DE 1, FAIRE

     /* Si deux éléments consécutifs sont mal rangés */
     SI T[compteur2] > T[compteur2 - 1]
       ALORS /* échange */
         temp = T[compteur2]
         T[compteur2] = T[compteur2 - 1]
         T[compteur2 - 1] = temp
     FIN SI

  FIN POUR
FIN POUR
Fin
```

2.5 Insertion dans une liste d'entiers triés

```
ALGORITHME d'insertion d'un élément dans une liste triée

Variables :
  T : tableau de N entiers triés (donné)
  Taille_max : entier (donné) /* taille max du tableau */
  valeur_insérée : entier (donnée)
  pos : entier
  compteur : entier

Début

/* Recherche de la position de l'élément à insérer */
pos = 0

TANT QUE T[pos] <  valeur_insérée et pos < Taille_max
   pos = pos +1
FIN TANT QUE

/* On décale les éléments placés à partir de la position pos */
POUR compteur ALLANT DE N à pos, PAR PAS DE 1 FAIRE
  T[compteur]=T[compteur-1]
FIN POUR

T[pos] = valeur_insérée

Fin
```