2020

Introduction à l'algorithmique

Travaux dirigés (TD)

Correction

Youssouf EL ALLIOUI
y.elallioui@usms.ma

Série 1

NOTIONS DE BASE

Exercice 1.1.

1) Quelles seront les valeurs des variables A et B après exécution des instructions suivantes ?

```
Variables A, B : Entier
Début
     A ← 1
     B ← A + 3
     A ← 3
Fin
```

Correction :

A = 3 et B = 4

2) Quelles seront les valeurs des variables A, B et C après exécution des instructions suivantes ?

```
Variables A, B, C : Entier
Début
     A ← 5
     B ← 3
     C ← A + B
     A ← 2
     C ← B – A
Fin
```

Correction :

A = 2, B = 3 et C = 1

3) Quelles seront les valeurs des variables A et B après exécution des instructions suivantes ?

```
Variables A, B : Entier
Début
     A ← 5
     B ← A + 4
     A ← A + 1
     B ← A – 4
Fin
```

Correction :

A = 6 et B = 2

4) Quelles seront les valeurs des variables A, B et C après exécution des instructions suivantes ?

```
Variables A, B, C : Entier
Début
     A ← 3
     B ← 10
     C ← A + B
     B ← A + B
     A ← C
Fin
```

Correction :

A = 13, B = 13 et C = 13

5) Quelles seront les valeurs des variables A et B après exécution des instructions suivantes ?

```
Variables A, B : Entier
Début
     A ← 5
     B ← 2
     A ← B
     B ← A
Fin
```

Correction :

A = 2 et B = 2

Questions : les deux dernières instructions permettent-elles d'échanger les deux valeurs de B et A ? Si l'on inverse les deux dernières instructions, cela change-t-il quelque chose ?

Correction :

Les deux dernières instructions ne permettent pas d'échanger les deux valeurs de B et A, puisque l'une des deux valeurs (celle de A) est ici écrasée.

Si l'on inverse les deux dernières instructions, cela ne changera rien du tout, hormis le fait que cette fois c'est la valeur de B qui sera écrasée.

6) Ecrire un algorithme permettant d'échanger les valeurs de deux variables A et B, et ce quel que soit leur contenu préalable.

Correction :

L'algorithme est :

```
Début
      C ← A
      A ← B
      B ← C
Fin
```

On est obligé de passer par une variable dite temporaire (la variable C).

7) On dispose de trois variables A, B et C. Ecrivez un algorithme transférant à B la valeur de A, à C la valeur de B et à A la valeur de C (toujours quels que soient les contenus préalables de ces variables).

Correction :

L'algorithme est :

```
Début
      D ← C
      C ← B
      B ← A
      A ← D
Fin
```

En fait, quel que soit le nombre de variables, une seule variable temporaire suffit.

8) Que produit l'algorithme suivant ?

```
Variables A, B, C : Chaines de Caractères
Début
      A ← "423"
      B ← "12"
      C ← A + B
Fin
```

Correction :

Il ne peut produire qu'une erreur d'exécution, puisqu'on ne peut pas additionner des caractères.

9) Que produit l'algorithme suivant ?

```
Variables A, B : Chaines de Caractères
Début
      A ← "423"
      B ← "12"
      C ← A & B
Fin
```

Correction :

On peut concaténer ces variables en utilisant l'opérateur &. A la fin de l'algorithme, C vaudra donc "42312".

Exercice 1.2.

Écrire un algorithme qui permet de déclarer deux variables A et B, saisir leurs valeurs, faire la somme et afficher le résultat ?

Correction :

```
Algo somme
     Variables a, b, s : Entiers
Début
     Ecrire ("Saisir la première valeur : ")
     Lire (a)
     Ecrire ("Saisir la deuxième valeur : ")
     Lire (b)
     s ← a + b
     Ecrire (a, " + ", b, " = ", s)
Fin
```

Exercice 1.3.

Écrire un algorithme qui permet de calculer et afficher la surface d'un cercle ?

Correction :

```
Algorithme surface_cercle
     Variables r, s : réel
     Constante PI ← 3.14 : réel
Début
     Ecrire ("Saisir le rayon : ")
     Lire (r)
     s ← r * r * PI
     Ecrire ("la surface = ", s)
Fin
```

Exercice 1.4.

Écrire un algorithme permettant d'échanger les valeurs de deux variables A et B, et ce quel que soit leur contenu préalable.

Correction :

```
Algorithme échange
     Variables a, b, c : entiers
Début
     Ecrire ("Saisir la valeur de A : ")
     Lire (a)
     Ecrire ("Saisir la valeur de B : ")
     Lire (b)
     c ← a
     a ← b
     b ← c
     Ecrire ("a = ", a, " b = ", b)
Fin
```

Exercice 1.5.

Écrire un algorithme qui permet de calculer et afficher le salaire brut d'un ouvrier connaissant le nombre d'heure et le tarif d'horaire ?

Correction :

```
Algorithme salaire_brut
     Variables salaire, tarif : réel
     Variables nb_heures : entier
Début
      Ecrire ("Saisir le tarif horaire : ")
      Lire (tarif)
      Ecrire ("Saisir le nombre d'heures : ")
      Lire (nb_heures)
      salaire ← tarif * nb_heures
      Ecrire ("Le salaire brut = ", salaire)
Fin
```

Exercice 1.6.

Écrire un algorithme qui fait la conversion d'une somme d'argent donnée en Euro, en une somme d'argent en DH ?

Correction :

```
Algorithme euro_dirham
      Variables s_euro, s_dh, taux_ech : réels
Début
      Ecrire ("Saisir la somme en Euro : ")
      Lire (s_euro)
      Ecrire ("Saisir le taux d'échange : ")
      Lire (taux_ech)
      s_dh ← s_euro * taux_ech
      Ecrire ("La somme en DH est : ", s_dh)
Fin
```

Exercice 1.7.

Ecrire un programme qui lit le prix HT d'un article, le nombre d'articles et le taux de TVA, et qui fournit le prix total TTC correspondant.

Correction :

```
Algorithme prix_TTC
      Variables nb_article : entiers
      Variables prix_HT, TVA, TTC : réels
Début
      Ecrire ("Saisir prix HT : ")
      Lire (prix_HT)
      Ecrire ("Saisir le nombre d'article : ")
      Lire (nb_article)
      Ecrire ("Saisir le taux TVA : ")
      Lire (TVA)
      TTC ← prix_ht * (1 + TVA)
```

```
     Ecrire ("le prix TTC est : ", TTC)
Fin
```

Exercice 1.8.

On dispose de trois variables A, B et C. Ecrivez un algorithme transférant à B la valeur de A, à C la valeur de B et à A la valeur de C.

Correction :

```
Algorithme Echange :
     Variables a, b, c, d : entiers
Début
     Ecrire ("Saisir la valeur de A : ")
     Lire (a)
     Ecrire ("Saisir la valeur de B : ")
     Lire (b)
     Ecrire ("Saisir la valeur de C : ")
     Lire (c)
     d ← a + b + c
     b ← d – b - c
     c ← d – b - c
     a ← d – b - c
     Ecrire ("A = ", a, " B = ", b, " C = ", c)
Fin
```

Exercice 1.9.

Ecrire l'algorithme permettant de saisir l'abscisse abs d'un point A et de calculer son ordonné $f(abs)$:

$$f(x) = 2\,x^2 - 3x + 4$$

Correction :

```
Algorithme Ordonné
     Variables abs, ord : réels
Début
     Ecrire ("Saisir l'abscisse du point : ")
     Lire (abs)
     ord ← 2 * (abs * abs) - 3 * abs + 4
     Ecrire ("l'ordonné du point est : ", ord)
Fin
```

Exercice 1.10.

Ecrire l'algorithme qui permet de saisir les paramètres d'une équation du second degré et de calculer son discriminant $delta$.

Correction :

```
Algorithme Discriminant
     Variables a, b, c, delta : entiers
Début
     Ecrire ("Saisir a : ")
     Lire (a)
```

```
     Ecrire ("Saisir b : ")
     Lire (b)
     Ecrire ("Saisir c : ")
     Lire (c)
     delta ← (b * b) - 4 * a * c
     Ecrire ("delta = ", delta)
Fin
```

Exercice 1.11.

Ecrire un programme qui demande deux nombres entiers à l'utilisateur, puis qui calcule et affiche la somme de ces nombres.

Correction :

```
Algorithme Somme_Deux_Nombres
     Variables A, B, SOMME : Entiers
Début
     Ecrire ("Entrez le premier nombre")
     Lire (A)
     Ecrire 'Entrez le deuxième nombre'
     Lire (B)
     SOMME ← A + B
     Ecrire ("La somme de ces deux nombres est : ")
     Ecrire (A, " + ", B, " = ", SOMME)
Fin
```

Exercice 1.12.

Le surveillant général d'un établissement scolaire souhaite qu'on lui écrit un programme qui calcule, pour chaque élève, la moyenne des notes des cinq matières. Ces matières sont avec leur coefficient :

Matière	Coefficient
Math	8
Physique	6
Français	4
Anglais	2
Histoire - Géographie	2

Correction :

```
Algorithme Moyenne_Des_Notes
Variables mat, phy, ang, fra, hg, moyenne : Réels
Début
     Ecrire ("Entrez la note de math : ")
     Lire (mat)
     Ecrire ("Entrez la note de physique : ")
     Lire (phy)
     Ecrire ("Entrez la note de français : ")
     Lire (fra)
     Ecrire ("Entrez la note d'anglais : ")
     Lire (ang)
     Ecrire ("Entrez la note d'histoire-Géo : ")
```

```
     Lire (hg)
     moyenne ← (mat * 8 + phy * 6 + fra * 4 + (ang + hg) * 2) / 22
     Ecrire ("La moyenne est : ", moyenne)
Fin
```

Exercice 1.13.

Ecrire un programme qui demande à l'utilisateur de taper la largeur et la longueur d'un champ et qui en affiche le périmètre et la surface.

Correction

```
Algorithme Périmètre_Surface
Variables largeur, longueur, périmètre, surface : Rèels
Début
     Ecrire ("Largeur : ")
     Lire (largeur)
     Ecrire ("Longueur : ")
     Lire (longueur)
     surface ← largeur * longueur
     périmètre ← 2 * (largeur + longueur)
     Ecrire ("Surface : ", surface)
     Ecrire ("Périmètre : ", périmètre)
Fin
```

Exercice 1.14.

Ecrire un programme qui demande à l'utilisateur de taper 5 entiers et qui affiche leur moyenne. Le programme ne devra utiliser que 2 variables.

Correction

```
Algorithme mouenne_deux_nombres
Variable a, b : réels
Début
     Ecrire ("Premier nombre : ")
     Lire (a)
     Ecrire ("Deuxième nombre : ")
     Lire (b)
     a ← (a + b)/2
     Ecrire ("Moyenne : ", a)
Fin
```

Exercice 1.15.

Ecrire un programme qui demande à l'utilisateur de taper le prix HT d'un kilo de tomates, le nombre de kilos de tomates achetés, le taux de TVA (Exemple $10\%, 20\%, ...$). Le programme affiche alors le prix TTC des marchandises.

Correction

```
Algorithme Calcul_Prix_TTC
Variable prixHT, nbrKilos, TVA, prixTTC : Rèels
Début
     Ecrire ("Prix HT d'un kilo de tomates : ")
```

```
    Lire (prixHT)
    Ecrire ("Nombre de kilos de tomates achetés : ")
    Lire (nbrKilos)
    Ecrire ("TVA : ")
    Lire (TVA)

    prixTTC ← prixHT * nbrKilos * (1 + TVA)

    Ecrire ("Prix TTC des marchandises : ", prixTTC)
Fin
```

Série 2

LES STRUCTURES ALTERNATIVES

Exercice 2. 1.

Ecrire l'algorithme qui permet de saisir un nombre puis déterminer s'il appartient à un intervalle donné, sachant que les extrémités de l'intervalle sont fixées par l'utilisateur.

Correction :

```
Algorithme Intervale
Variables nb, a, b : Entiers
Début
     Ecrire ("Saisir un nombre")
     Lire (nb)
     Ecrire ("Saisir l'extremité de l'intervalle")
     Lire (a, b)
     SI (nb <= b ET nb >= a) ALORS
          Ecrire (a, " appartient à l'intervalle ")
     SINON
          Ecrire (a, " n'appartient pas à l'intervalle")
     FIN SI
FIN
```

Exercice 2. 2.

Ecrire l'algorithme qui permet de saisir le jour, le mois et l'année d'une date (Mois : numéro du mois), et de déterminer si elle est correcte ou non, et ou est l'erreur. on suppose que :

- l'année doit être entre 1900 et 2019
- le mois entre 1 et 12
- le jour est entre 1 et 30

Correction :

```
Algorithme valider_date
Variables annee, mois, jour : Entiers
Début
     Ecrire ("Saisir une année")
     Lire (annee)
     Ecrire ("Saisir le mois")
     Lire (mois)
     Ecrire ("Saisir le jour")
     Lire (jour)
     SI (annee < 1900 OU annee > 2019) ALORS
          Ecrire ("l'année est incorrecte")
     SINON SI (mois < 1 OU mois > 12) ALORS
          Ecrire ("le mois est incorrect")
```

```
     SINON SI (jour < 1 OU jour > 30) ALORS
          Ecrire ("le jour est incorrect")
     FIN SI
     SINON
          Ecrire ("la date est correcte")
     FIN SI
FIN
```

Exercice 2. 3.

Ecrire un algorithme qui demande deux nombres à l'utilisateur et l'informe ensuite si leur produit est négatif ou positif. Attention toutefois : on ne doit pas calculer le produit des deux nombres.

Correction :

```
Algorithme signe_produit
Variables a, b : entiers
Début
     Ecrire ("Saisir deux nombres")
     Lire (a, b)
     SI ((a > 0 ET b > 0) OU (a < 0 ET b < 0)) ALORS
          Ecrire ("Le produit est positif")
     SINON
          Ecrire ("Le produit est négatif")
     FIN SI
FIN
```

Exercice 2. 4.

Ecrire l'algorithme qui permet de calculer le montant des heures supplémentaires d'un employé, sachant le prix unitaire d'une heure selon le barème suivant :

- Les 39 premières heures sans supplément,
- De la 40ième à la 44ième heure sont majorées de 50%,
- De la 45ième à la 49ième heure sont majorées de 75%,
- De la 50ième heure ou plus, sont majorées de 100%.

Correction :

```
Algorithme heures_supp
Variables prix_heure, montant_supp : réels
Variables nb_heures : entier
Début
     Ecrire ("Saisir le nombre d'heures")
     Lire (nb_heure)
     Ecrire ("Saisir le prix par heure")
     Lire (prix_heure)
     SI (nb_heures <= 39) ALORS
          montant_sup ← 0
     SINON
          SI (nb_heures <= 44) ALORS
               montant_supp ← (nb_heures - 39) * prix_heure * 1.5
        SINON SI (nb_heures <= 49) ALORS
```

```
            montant_supp ← (5 * prix_heure * 1.5) + ((nb_heures - 44) * prix_heure
* 1.75)
        SINON
            montant_supp ← (5 * prix_heure * 1.5) + (5 * prix_heure * 1.75) +
((nb_heures - 49) * prix_heure * 2)
        FIN SI
     FIN SI
          Ecrire("Le montant des heures supplémentaires est : ",montant_supp)
FIN
```

Exercice 2. 5.

Ecrivez un algorithme qui lira au clavier l'heure et les minutes, et il affichera l'heure qu'il sera une minute plus tard.

Par exemple, si l'utilisateur tape 21 puis 32, l'algorithme doit répondre : "Dans une minute, il sera 21 heure(s) 33".

NB : on suppose que l'utilisateur entre une heure valide. Pas besoin donc de la vérifier.

Correction :

```
Algorithme heure_a_minute
Variables heure, minute : entiers
Début
     Ecrire ("Saisir l'heure et la minute : ")
     Lire (heure, minute)
     SI (heure = 23) ALORS
          SI (minute = 59) ALORS
              heure ← 0
              minute ← 0
          SINON
              minute ← minute+1
          FIN SI
     SINON
          SI (minute = 59) ALORS
              heure ← heure+1
              minute ← 0
          SINON
              minute ← minute+1
          FIN SI
     FIN SI
     Ecrire ("Dans une minute, il sera ", heure, " heure(s) ", minute)
Fin
```

Exercice 2. 6.

Vous gérez les stocks et les commandes d'une entreprise. Vous ne pouvez pas honorer une commande si vous n'avez pas la totalité de la quantité demandée. Donc créer un algorithme permettant d'afficher la quantité que l'entreprise doit livrer au client. Si votre stock est inférieur, vous ne pouvez livrer que la quantité que vous possédez.

Correction :

```
Algorithme quantite_livrer
Variables qte_stock, qte_demandee : entiers
Début
     Ecrire ("Saisir la quantité en stock")
```

```
    Lire (qte_stock)
    Ecrire ("Saisir la quantité demandée")
    Lire (qte_demandee)
    SI (qte_demandee <= qte_stock) ALORS
        Ecrire ("la quantité à livrer est : ", qte_demandee)
    SINON
        Ecrire ("la quantité à livrer est : ", qte_stock)
    FIN SI
FIN
```

Exercice 2. 7.

Ecrire un algorithme qui demande deux nombres à l'utilisateur et l'informe ensuite si le produit est négatif ou positif (on inclut cette fois le traitement du cas où le produit peut être nul). Attention toutefois, on ne doit pas calculer le produit !

Correction

```
Algorithme Produit_Négatif_Positif
Variables m, n : Entier
Début
     Ecrire ("Entrez deux nombres : ")
     Lire m, n
     Si (m = 0 OU n = 0) Alors
          Ecrire ("Le produit est nul")
     Sinon Si ((m < 0 ET n < 0) OU (m > 0 ET n > 0)) Alors
          Ecrire ("Le produit est positif")
     Sinon
          Ecrire ("Le produit est négatif")
     Fin si
Fin
```

Exercice 2. 8.

Ecrire un algorithme qui demande l'âge d'un enfant à l'utilisateur. Ensuite, il l'informe de sa catégorie :

- « Poussin » de 6 à 7 ans
- « Pupille » de 8 à 9 ans
- « Minime » de 10 à 11 ans
- « Cadet » après 12 ans

Correction

```
Algorithme Age_Catégorie
Variable age : Entier
Début
     Ecrire ("Entrez l'âge de l'enfant : ")
     Lire age
     Si (age >= 12) Alors
          Ecrire ("Catégorie Cadet")
     Sinon Si (age >= 10) Alors
          Ecrire ("Catégorie Minime")
     Sinon Si (age >= 8) Alors
          Ecrire ("Catégorie Pupille")
```

```
     Sinon Si (age >= 6) Alors
          Ecrire ("Catégorie Poussin")
     Finsi
Fin
```

Exercice 2. 9.

Écrire un algorithme qui, à partir d'un nombre compris entre 1 et 7, affiche le jour correspondant ?

Correction :

```
Algorithme jours_semaine
Variable jour : entier ;
Début
     Ecrire ("Saisir une valeur : ")
     Lire (jour)
     SELON jour faire
          1 : Ecrire ("Lundi")
          2 : Ecrire ("Mardi")
          3 : Ecrire ("Mercredi")
          4 : Ecrire ("Jeudi")
          5 : Ecrire ("Vendredi")
          6 : Ecrire ("Samedi")
          7 : Ecrire ("Dianche")
          SINON : Ecrire("Jour invalide")
     Fin SELON
Fin
```

Exercice 2. 10.

Écrire un algorithme qui à partir d'une note affiche la mention correspondante ?

Correction :

```
Algorithme mention
Variable note : réel
Début
     Ecrire ("Saisir une note : ")
     Lire(note)
     SELON A faire
          note < 10 : Ecrire("Non admin")
          note < 12 : Ecrire("passable")
          note < 14 : Ecrire("assez bien")
          note < 16 : Ecrire("Bien")
          note < 18 : Ecrire("très bien")
          note <= 20 : Ecrire("Excellent")
          SINON : Ecrire ("Note invalide")
     Fin SELON
Fin
```

Exercice 2. 11.

Ecrire un algorithme qui, à partir de trois variables a, b et c de R, calcule les solutions de l'équation de la forme suivante :

$$(E) : \ \boldsymbol{ax^2 + bx + c = 0} \qquad a, b \text{ et } c \text{ sont des réels}$$

On prendra garde à bien tester tous les cas possibles :

- a est nul, et l'équation est en fait une équation du premier degré. Exemple : $4x - 2 = 0$ donne une unique solution : $x = 0.5$.
- Le discriminant $\Delta = (b^2 - 4ac)$ est nul, et il n'y a qu'une seule solution, appelée racine double, au problème.

  Exemple : $2x^2 + 4x + 2 = 0$ donne la solution : $x = -1$.
- Le discriminant Δ est positif, et deux solutions existent : $x1 = (-b - \Delta 1/2)/2a$ et $x2 = (-b + \Delta 1/2)/2a$.
  Exemple : $2x^2 + x - 6 = 0$ donne $x1 = 1.5$ et $x2 = -2$.
- Le discriminant Δ est négatif, et il n'existe pas de solutions (réelles) au problème.

  Exemple : $x^2 + 1 = 0$ n'admit pas de solution dans R.

Correction :

```
Algorithme mention
Variable a, b, c, delta, x1, x2 : réels
Début
     Ecrire ("Entrer a : ")
     Lire (a)
     Ecrire ("Entrer b : ")
     Lire (b)
     Ecrire ("Entrer c : ")
     Lire (c)

     SI (a = 0)
          SI (b = 0)
               SI (c = 0)
                    Ecrire ("S = R")
               SINON
                    Ecrire ("Pas de solution !")
          SINON
               x1 ← -c/b
               Ecrire ("S = {", x1, "}")
     SINON
          delta ← b^2 − 4 * a * c
          SI (delta < 0)
               Ecrire ("Pas de solution !")
          SINON SI (delta = 0)
               x1 ← -b/2*a
               Ecrire ("S = {", x1, "}")
          SINON
               X1 ← (−b − √delta) / (2*a)
               X2 ← (−b + √delta) / (2*a)
               Ecrire ("S = {", x1, ", ", x2, "}")
Fin
```

Série 3

LES STRUCTURES REPETITIVES

Exercice 3. 1.

Ecrire un algorithme qui demande à l'utilisateur un nombre compris entre 1 et 3 jusqu'à ce que la réponse convienne.

Correction :

Voir le **support du cours** → partie **5.2 La structure TANT QUE** → **Exercices** → **1.**

Exercice 3. 2.

Ecrire un algorithme qui demande un nombre compris entre 10 et 20, jusqu'à ce que la réponse convienne. En cas de réponse supérieure à 20, on fera apparaître un message : « Plus petit ! », et inversement, « Plus grand ! » si le nombre est inférieur à 10.

Correction :

Voir le **support du cours** → partie **5.2 La structure TANT QUE** → **Exercices** → **2.**

Exercice 3. 3.

Ecrire un algorithme qui demande un nombre de départ, et qui ensuite affiche les dix nombres suivants. Par exemple, si l'utilisateur entre le nombre 17, le programme affichera les nombres de 18 à 27.

Correction :

Voir le **support du cours** → partie **5.2 La structure TANT QUE** → **Exercices** → **3.**

Exercice 3. 4.

Ecrire un algorithme qui demande un nombre de départ, et qui ensuite écrit la table de multiplication de ce nombre, présentée comme suit (cas où l'utilisateur entre le nombre 7) :

Table de 7 :

```
7 x 1 = 7
7 x 2 = 14
7 x 3 = 21
…
7 x 10 = 70
```

Correction :

Voir le **support du cours** → partie **5.1 La structure POUR** → **Exercices** → **1.**

Exercice 3. 5.

Ecrire un algorithme qui demande un nombre de départ, et qui calcule la somme des entiers jusqu'à ce nombre. Par exemple, si l'on entre 5, le programme doit calculer :

```
1 + 2 + 3 + 4 + 5 = 15
```

NB : on souhaite afficher uniquement le résultat, pas la décomposition du calcul.

Correction :

Voir le **support du cours** → partie **5.1 La structure POUR** → **Exercices** → **2.**

Exercice 3. 6.

Ecrire un programme qui demande de saisir un entier N et qui affiche $N!$:

$$N! = N * (N-1) * (N-2) * \dots * 3 * 2 * 1$$

Exemple : la factorielle de 8 :

$$\mathbf{8! = 1 \times 2 \times 3 \times 4 \times 5 \times 6 \times 7 \times 8}$$

Correction :

Voir le **support du cours** → partie **5.1 La structure POUR** → **Exercices** → **3.**

Exercice 3. 7.

Ecrire un algorithme qui demande successivement 20 nombres à l'utilisateur, et qui lui dise ensuite quel était le plus grand parmi ces 20 nombres :

```
Entrez le nombre numéro 1 : 12
Entrez le nombre numéro 2 : 14
...
Entrez le nombre numéro 20 : 6
Le plus grand de ces nombres est : 14
```

Modifiez ensuite l'algorithme pour que le programme affiche de surcroît en quelle position avait été saisie ce nombre :

```
C'était le nombre numéro 2
```

Correction :

Voir le **support du cours** → partie **5.1 La structure POUR** → **Exercices** → **4.**

Exercice 3. 8.

Réécrire l'algorithme précédent, mais cette fois-ci on ne connaît pas d'avance combien l'utilisateur souhaite saisir de nombres. La saisie des nombres s'arrête lorsque l'utilisateur entre un zéro.

Correction :

```
Variables N, i, PG, IPG : Entier
Debut
N ← 1
i ← 0
PG ← 0
TantQue N <> 0
  Ecrire "Entrez un nombre : "
  Lire N
  i ← i + 1
  Si i = 1 ou N > PG Alors
    PG ← N
    IPG ← i
  FinSi
FinTantQue
Ecrire "Le nombre le plus grand était : ", PG
Ecrire "Il a été saisi en position numéro ", IPG
Fin
```

Exercice 3. 9.

Lire la suite des prix (en euros entiers et terminée par zéro) des achats d'un client. Calculer la somme qu'il doit, lire la somme qu'il paye, et simuler la remise de la monnaie en affichant les textes "10 Euros", "5 Euros" et "1 Euro" autant de fois qu'il y a de coupures de chaque sorte à rendre.

Correction :

```
Variables E, somdue, M, Reste, Nb10E, Nb5E : Entier
Debut
E ← 1
somdue ← 0
TantQue E <> 0
  Ecrire "Entrez le montant : "
  Lire E
  somdue ← somdue + E
FinTantQue
Ecrire "Vous devez :", somdue, " euros"
Ecrire "Montant versé :"
Lire M
Reste ← M - somdue
Nb10E ← 0
TantQue Reste >= 10
  Nb10E ← Nb10E + 1
  Reste ← Reste – 10
FinTantQue
Nb5E ← 0
Si Reste >= 5
```

```
  Nb5E ← 1
  Reste ← Reste – 5
FinSi
Ecrire "Rendu de la monnaie :"
Ecrire "Billets de 10 E : ", Nb10E
Ecrire "Billets de  5 E : ", Nb5E
Ecrire "Pièces de 1 E : ", reste
Fin
```

Exercice 3. 10.

Écrire un algorithme qui permette de connaître ses chances de gagner au tiercé, quarté, quinté et autres impôts volontaires.

On demande à l'utilisateur le nombre de chevaux partants, et le nombre de chevaux joués. Les deux messages affichés devront être :

```
Dans l'ordre : une chance sur X de gagner
Dans le désordre : une chance sur Y de gagner
```

X et Y nous sont donnés par la formule suivante, si n est le nombre de chevaux partants dans l'exercice 5.6 ci-dessus) :

```
X = n ! / (n - p) !
Y = n ! / (p ! * (n – p) !)
```

NB : cet algorithme peut être écrit d'une manière simple, mais relativement peu performante. Ses performances peuvent être singulièrement augmentées par une petite astuce. Vous commencerez par écrire la manière la plus simple, puis vous identifierez le problème, et écrirez une deuxième version permettant de le résoudre.

Correction :

Spontanément, on est tenté d'écrire l'algorithme suivant :

```
Variables N, P, i, Numé, Déno1, Déno2 : Entier
Debut Ecrire "Entrez le nombre de chevaux partants : "
Lire N
Ecrire "Entrez le nombre de chevaux joués : "
Lire P
Numé ← 1
Pour i ← 2 à N
  Numé ← Numé * i
Fin Pour
Déno1 ← 1
Pour i ← 2 à N-P
  Déno1 ← Déno1 * i
Fin Pour
Déno2 ← 1
Pour i ← 2 à P
  Déno2 ← Déno2 * i
Fin Pour
Ecrire "Dans l'ordre, une chance sur ", Numé / Déno1
Ecrire "Dans le désordre, une sur ", Numé / (Déno1 * Déno2)
Fin
```

Cette version, formellement juste, comporte tout de même deux faiblesses.

La première, et la plus grave, concerne la manière dont elle calcule le résultat final. Celui-ci est le quotient d'un nombre par un autre ; or, ces nombres auront rapidement tendance à être très grands. En calculant, comme on le fait ici, d'abord le numérateur, puis ensuite le dénominateur, on prend le risque de demander à la machine de stocker des nombres trop grands pour qu'elle soit capable de les coder (cf. le préambule). C'est d'autant plus bête que rien ne nous oblige à procéder ainsi : on n'est pas obligé de passer par la division de deux très grands nombres pour obtenir le résultat voulu.

La deuxième remarque est qu'on a programmé ici trois boucles successives. Or, en y regardant bien, on peut voir qu'après simplification de la formule, ces trois boucles comportent le même nombre de tours ! (si vous ne me croyez pas, écrivez un exemple de calcul et biffez les nombres identiques au numérateur et au dénominateur). Ce triple calcul (ces trois boucles) peut donc être ramené(es) à un(e) seul(e). Et voilà le travail, qui est non seulement bien plus court, mais aussi plus performant :

```
Variables N, P, i, A, B : Entiers
Debut
Ecrire "Entrez le nombre de chevaux partants : "
Lire N
Ecrire "Entrez le nombre de chevaux joués : "
Lire P
A ← 1
B ← 1
Pour i ← 1 à P
  A ← A * (i + N - P)
  B ← B * i
Fin Pour
Ecrire "Dans l'ordre, une chance sur ", A
Ecrire "Dans le désordre, une chance sur ", A / B
Fin
```

Exercice 3. 11.

Ecrire un programme qui demande à l'utilisateur de taper un entier N et qui calcule u(N) défini par :

$$u(0) = 3$$
$$u(n+1) = 3.u(n) + 4$$

Correction :

```
Variables i, u=3, N : Entiers
Debut
    Ecrire ("Tapez N : ")
    Lire (N)
    Pour i=0 A N Faire
      U = u*3 + 4
    Fin Pour
    Ecrire ("u(", N, ") = ", u
Fin
```

Exercice 3. 12.

Ecrire un programme qui demande à l'utilisateur de taper un entier N et qui calcule u(N) défini par :

$$u(0) = 1$$
$$u(1) = 1$$
$$u(n+1) = u(n) + u(n-1)$$

Correction :

```
Variables i, u=1, v=1, w, N : Entiers
Debut
    Ecrire ("Tapez N : ")
    Lire (N)

    Pour i=2 A N Faire
       w=u+v
       u=v
       v=w
    Fin Pour
    Ecrire ("u(", N, ") = ", w
Fin
```

Exercice 3. 13.

Ecrire un programme qui permet de faire des opérations sur un entier (valeur initiale à 0). Le programme affiche la valeur de l'entier puis affiche le menu suivant :

1. Ajouter
2. Multiplier par
3. Soustraire
4. Quitter

Le programme demande alors de taper un entier entre 1 et 4. Si l'utilisateur tape une valeur entre 1 et 3, on effectue l'opération correspondante, on affiche la nouvelle valeur de l'entier puis on réaffiche le menu et ainsi de suite jusqu'à ce qu'on tape 4. Lorsqu'on tape 4, le programme se termine.

Correction :

```
Variables x=0, choix : Entier
Debut
    Répéter
        Ecrire ("1 : Ajouter 1")
        Ecrire ("2 : Multiplier par 2")
        Ecrire ("3 : Soustraire 4")
        Ecrire ("4 : Quitter")
        Ecrire ("Votre choix : ")
        Lire (choix)

        Selon (choix)
             cas 1 : x++
             break
             cas 2: x=x*2
```

```
            break
            cas 3: x=x-4
            break;
        Fin Selon
    Jusqu'à (choix != 4)

    Ecrire ("La valeur finale de x vaut : ", x)
Fin
```

Exercice 3. 14.

Ecrire un programme qui demande à l'utilisateur de taper des entiers strictement positifs et qui affiche leur moyenne. Lorsqu'on tape une valeur négative, le programme affiche ERREUR et demande de retaper une valeur. Lorsqu'on tape 0, cela signifie que le dernier entier a été tapé. On affiche alors la moyenne. Si le nombre d'entiers tapés est égal à 0, on n'affiche PAS DE MOYENNE.

Correction :

```
Variables x, s=0, nb=0 : Entier
Variable moyenne : Reel
Debut
    Répéter
        Ecrire ("Tapez un entier :")
        Lire (x)

        Si (x>0) Alors
             s = s+x
             nb++
        Sinon Si (x<0) Alors
             Ecrire ("ERREUR !")
        Fin Si
    Jusqu'à (x != 0)

    Si (nb=0) Alors
        Ecrire ("AUCUN ENTIER TAPE ")
        Ecrire ("PAS DE MOYENNE")
    Sinon
        Moyenne = (double)s/nb
        Ecrire ("La moyenne vaut : ", moyenne)
    Fin Si

Fin
```

Série 4

LES TABLEAUX

Exercice 4. 1.

Écrire un programme qui demande à l'utilisateur de saisir 10 entiers stockés dans un tableau. Le programme doit afficher le nombre d'entiers supérieurs ou égaux à 10.

Correction :

```
Tableau t[10] : Entier
Variables i, nb=0 : Entier
Debut
    Pour i=0 A 10 Faire
        Ecrire ("Tapez un entier ")
        Lire (t[i])
    Fin Pour

    Pour i=0 A 10 Faire
        Si (t[i]>10) Alors
             Nb++
        Fin de Si
    Fin Pour

    Ecrire ("Le nombre d'entiers valant 10 ou davantage est : ", nb)
Fin
```

Exercice 4. 2.

Ecrire un programme qui demande à l'utilisateur de saisir 10 entiers stockés dans un tableau ainsi qu'un entier V. Le programme doit rechercher si V se trouve dans le tableau et afficher "V se trouve dans le tableau" ou "V ne se trouve pas dans le tableau".

Correction :

```
Tableau t[10] : Entier
Variables i, v : Entier
Variable trouve : booléen
Debut
    Pour i=0 A 10 Faire
        Ecrire ("Tapez un entier ")
        Lire (t[i])
    Fin Pour

    Ecrire ("Tapez la valeur de V : ")
    Lire (v)

    trouve = False
    i = 0

    Pour i=0 A 10 Faire
        Si (t[i]=v) Alors
             trouve = True
```

```
        Fin de Si
    Fin Pour

    Si (trouve) Alors
        Ecrire ("La valeur ", V, " se trouve dans le tableau")
    Else
        Ecrire ("La valeur ", V, " ne se trouve pas dans le tableau")
    Fin Si
Fin
```

Exercice 4. 3.

Ecrire un programme qui demande à l'utilisateur de saisir 10 entiers stockés dans un tableau. Le programme doit ensuite afficher l'indice du plus grand élément.

Correction :

```
Tableau t[10] : Entier
Variables i, indice : Entiers
Debut
    Pour i=0 A 9 Faire
        Ecrire ("Tapez un entier ")
        Lire (t[i])
    Fin Pour

    indice ← 0

    Pour i=0 A 9 Faire
        Si (t[indice]<t[i]) Alors
            indice ← i
        Fin Si
    Fin Pour

    Ecrire ("L'indice du plus grand élément est : ", indice)
Fin
```

Exercice 4. 4.

Ecrire un programme qui demande à l'utilisateur de saisir 10 entiers stockés dans un tableau ainsi qu'un entier V. Le programme doit rechercher si V se trouve dans le tableau et doit supprimer la première occurrence de V en décalant d'une case vers la gauche les éléments suivants et en rajoutant un 0 à la fin du tableau. Le programme doit ensuite afficher le tableau final.

Correction :

```
Tableau t[10] : Entier
Variables i, j, v : Entiers
Variable trouve : Booléen
Debut
    Pour i=0 A 9 Faire
        Ecrire ("Tapez un entier ")
        Lire (t[i])
    Fin Pour

    Ecrire ("Tapez la valeur de V : ")
```

```
    Lire (v)

    trouve ← NON
    i ← 0

    Tant que (!trouve ET i<N)
        Si (t[i]=V) Alors
            trouve ← OUI
        Sinon
            i++
        Fin Si
    Fin Tant que

    Si (trouve) Alors
        Pour j=i A 9 Faire
            t[j] ← t[j+1]
        Fin Pour
        t[9-1] ← 0
    Fin Si

    Pour i=0 A 9 Faire
        Ecrire (t[i])
    Fin Pour
Fin
```

Exercice 4. 5.

Ecrire un programme qui lit la dimension N d'un tableau T du type ***int***, remplit le tableau par des valeurs entrées au clavier et affiche le tableau.

Calculer et afficher ensuite la somme des éléments du tableau.

Correction :

```
Variables I, N, SOM : Entiers
Debut
    //Saisie des données:
    Ecrire ("Dimension du tableau (max.50) : ")
    Lire (N)

    // Création du tableau sur la base de la valeur de N :
    Tableau T[N] : Entier

    // Remplissage du tableau :
    Pour I=0 A N Faire
        Ecrire ("Tapez un entier ")
        Lire (t[I])
    Fin Pour

    // Affichage du tableau :
    Pour I=0 A N Faire
        Ecrire (T[I])
    Fin Pour

    // Calcul de la somme :
    SOM ← 0
```

```
    Pour I=0 A N Faire
        SOM ← SOM + T[I]
    Fin Pour

    // Edition du résultat :
    Ecrire ("Somme de éléments : ", SOM)
Fin
```

Exercice 4. 6.

Ecrire un programme qui lit la dimension *N* d'un tableau *T* du type *int*, remplit le tableau par des valeurs entrées au clavier et affiche le tableau.

Effacer ensuite toutes les occurrences de la valeur 0 dans le tableau *T* et tasser les éléments restants. Afficher le tableau résultant.

Correction :

```
Variables I, J, N : Entiers
Debut
    //Saisie des données:
    Ecrire ("Dimension du tableau (max.50) : ")
    Lire (N)

    // Création du tableau sur la base de la valeur de N :
    Tableau T[N] : Entier

    // Remplissage du tableau :
    Pour I=0 A N Faire
        Ecrire ("Tapez un entier ")
        Lire (t[I])
    Fin Pour

    // Affichage du tableau :
    Pour I=0 A N Faire
        Ecrire (T[I])
    Fin Pour

    /*
    Effacer les zéros et comprimer : Copier tous les éléments de I vers
    J et augmenter J pour les éléments non nuls.
    */
    J ← 0
    Pour I=0 A N Faire
        T[J] ← T[I]
        Si (T[I]) Alors
            J++
        Fin Si
    Fin Pour

    // Nouvelle dimension du tableau :
    N ← J

    // Edition du résultat :
    Ecrire ("Tableau résultat : ")
    Pour I=0 A N Faire
```

```
        Ecrire (T[I])
    Fin Pour
Fin
```

Exercice 4. 7.

Ecrire un programme qui lit la dimension *N* d'un tableau *T* du type *int*, remplit le tableau par des valeurs entrées au clavier et affiche le tableau.

Ranger ensuite les éléments du tableau *T* dans l'ordre inverse sans utiliser de tableau d'aide. Afficher le tableau résultant.

Idée : Echanger les éléments du tableau à l'aide de deux indices qui parcourent le tableau en commençant respectivement au début et à la fin du tableau et qui se rencontrent en son milieu.

Correction :

```
Variables I, J, N, AIDE : Entiers
Debut
    //Saisie des données:
    Ecrire ("Dimension du tableau (max.50) : ")
    Lire (N)

    // Création du tableau sur la base de la valeur de N :
    Tableau T[N] : Entier

    // Remplissage du tableau :
    Pour I=0 A N Faire
        Ecrire ("Tapez un entier ")
        Lire (t[I])
    Fin Pour

    // Affichage du tableau :
    Pour I=0 A N Faire
        Ecrire (T[I])
    Fin Pour

    // Inverser le tableau :
    J ← N-1
    Pour I=0 A J Faire
        // Echange de T[I] et T[J] :
        AIDE ← T[I]
        T[I] ← T[J]
        T[J] ← AIDE
    Fin Pour

    // Edition du résultat :
    Ecrire ("Tableau résultat : ")
    Pour I=0 A N Faire
        Ecrire (T[I])
    Fin Pour
Fin
```

Exercice 4. 8.

Ecrire un programme qui lit la dimension *N* d'un tableau statique *T* du type *int*, remplit le tableau par des valeurs entrées au clavier et affiche le tableau.

Copiez ensuite toutes les composantes strictement positives dans un deuxième tableau dynamique $TPOS$ et toutes les valeurs strictement négatives dans un troisième tableau dynamique $TNEG$.

Afficher les tableaux $TPOS$ et $TNEG$.

Correction :

```
Variables I, N, NPOS, NNEG : Entiers
Tableau TPOS(), TNEG() : Entiers
Debut
    //Saisie des données:
    Ecrire ("Dimension du tableau (max.50) : ")
    Lire (N)

    // Création du tableau sur la base de la valeur de N :
    Tableau T[N] : Entier

    // Remplissage du tableau :
    Pour I=0 A N Faire
        Ecrire ("Tapez un entier ")
        Lire (t[I])
    Fin Pour

    // Affichage du tableau :
    Pour I=0 A N Faire
        Ecrire (T[I])
    Fin Pour

    // Initialisation des dimensions de TPOS et TNEG
    NPOS ← 0
    NNEG ← 0

    //Transfert des données :
    Pour I=0 A N-1 Faire
        Si (T[I]>0) Alors
             NPOS++
             Redim TPOS[NPOS]
             TPOS[NPOS] ← T[I]
        Fin Si
        Si (T[I]<0) Alors
             NNEG++
             Redim TNEG[NNEG]
             TNEG[NNEG] ← T[I]
        Fin Si
    Fin Pour

    // Edition du résultat :
    Ecrire ("Tableau TPOS :\n")
    Pour I=0 A NPOS-1 Faire
        Ecrire (TPOS[I])
    Fin Pour
    Ecrire ("Tableau TNEG :\n")
    Pour I=0 A NNEG-1 Faire
        Ecrire (TNEG[I])
```

```
    Fin Pour
Fin
```

Exercice 4. 9.

Ecrire un programme qui lit les dimensions L et C d'un tableau T à deux dimensions du type int. Remplir le tableau par des valeurs entrées au clavier et afficher le tableau ainsi que la somme de tous ses éléments.

Correction :

```
Variables I, J, L, C, SOM : Entiers
Début
    //Saisie des données:
    Ecrire ("Nombre de lignes (max.50) : ")
    Lire (L)
    Ecrire ("Nombre de colonnes (max.50) : ")
    Lire (C)

    // Création du tableau sur la base des valeurs de L et C :
    Tableau T[L][C] : Entier

    // Remplissage du tableau :
    Pour I=1 A L Faire
        Pour J=1 A C Faire
            Ecrire ("Tapez un entier ")
            Lire (t[I][J])
        Fin Pour
    Fin Pour

    // Affichage du tableau :
    Pour I=1 A L Faire
        Pour J=1 A C Faire
            Ecrire (t[I][J])
        Fin Pour
    Fin Pour

    // Calcul de la somme
    SOM ← 0
    Pour I=1 A L Faire
        Pour J=1 A C Faire
            SOM += T[I][J]
        Fin Pour
    Fin Pour

    // Edition du résultat :
    Ecrire (""Somme des éléments : ", SOM)
Fin
```

Exercice 4. 10.

Ecrire un programme qui lit les dimensions L et C d'un tableau T à deux dimensions du type int. Remplir le tableau par des valeurs entrées au clavier et afficher le tableau ainsi que la somme de chaque ligne et de chaque colonne en n'utilisant qu'une variable d'aide pour la somme.

Correction :

```
Variables I, J, L, C, SOM : Entiers
Début
    //Saisie des données:
    Ecrire ("Nombre de lignes (max.50) : ")
    Lire (L)
    Ecrire ("Nombre de colonnes (max.50) : ")
    Lire (C)

    // Création du tableau sur la base des valeurs de L et C :
    Tableau T[L][C] : Entier

    // Remplissage du tableau :
    Pour I=1 A L Faire
        Pour J=1 A C Faire
            Ecrire ("Tapez un entier ")
            Lire (t[I][J])
        Fin Pour
    Fin Pour

    // Affichage du tableau :
    Pour I=1 A L Faire
        Pour J=1 A C Faire
            Ecrire (t[I][J])
        Fin Pour
    Fin Pour

    // Calcul et affichage de la somme des lignes :
    Pour I=1 A L Faire
        SOM ← 0
        Pour J=1 A C Faire
            SOM += T[I][J]
        Fin Pour
        Ecrire ("Somme des éléments de la ligne ", I, " est : ", SOM)
    Fin Pour

    // Calcul et affichage de la somme des lignes :
    Pour J=1 A C Faire
        SOM ← 0
        Pour I=1 A L Faire
            SOM += T[I][J]
        Fin Pour
        Ecrire ("Somme des éléments de la colonne ", J, " est : ", SOM)
    Fin Pour
Fin
```

Exercice 4. 11. Maximum et minimum des valeurs d'un tableau

Ecrire un programme qui détermine la plus grande et la plus petite valeur dans un tableau d'entiers *A*. Afficher ensuite la valeur et la position du maximum et du minimum. Si le tableau contient plusieurs maximums ou minimums, le programme retiendra la position des premiers rencontrés.

Correction :

```
Variables I, N, MIN, MAX : Entiers
```

```
Debut
    //Saisie des données:
    Ecrire ("Dimension du tableau (max.50) : ")
    Lire (N)

    // Création du tableau sur la base de la valeur de N :
    Tableau A[N] : Entier

    // Remplissage du tableau :
    Pour I=1 A N Faire
        Ecrire ("Tapez un entier ")
        Lire (A[I])
    Fin Pour

    // Affichage du tableau :
    Pour I=1 A N Faire
        Ecrire (A[I])
    Fin Pour

    // Recherche du maximum et du minimum :
    MIN ← 0
    MAX ← 0
    Pour I=1 A N Faire
        Si(A[I]> A[MAX]) Alors
            MAX ← I
        Fin Si
        Si(A[I]< A[MIN]) Alors
            MIN ← I
        Fin Si
    Fin Pour

    // Edition du résultat :
    Ecrire ("Position du minimum : ", MIN)
    Ecrire ("Position du maximum : ", MAX)
    Ecrire ("Valeur du minimum : ", A[MIN])
    Ecrire ("Valeur du maximum : ", A[MAX])
Fin
```

Exercice 4. 12. Insérer une valeur dans un tableau trié

Un tableau dynamique A de dimension N, contient N valeurs entières triées par ordre croissant ; Ecrire un programme permettant d'insérer une valeur VAL, donnée au clavier, dans le tableau A de manière à obtenir un tableau de $N + 1$ valeurs triées.

Correction :

```
Variables I, N, VAL : Entiers
Tableau A() : Entier        // Tableau dynamique
Debut
    //Saisie des données:
    Ecrire ("Dimension initiale du tableau A : ")
    Lire (N)

    // Redimensionnement du tableau A sur la base de la valeur de N :
    Redim A[N]
```

```
    // Remplissage du tableau :
    Pour I=1 A N Faire
        Ecrire ("Tapez un entier ")
        Lire (A[I])
    Fin Pour

    // Affichage du tableau :
    Pour I=1 A N Faire
        Ecrire (A[I])
    Fin Pour

    // Saisie de VAL :
    Ecrire ("Elément à insérer : ")
    Lire (VAL)

    // Ajouter une case mémoire au tableau A
    N ← N+1
    Redim A[N]

    /* Déplacer les éléments plus grands que VAL d'une position vers
    l'arrière. */
    Pour I=N-1 A 1 Pas de -1 Faire
        Si (A[I]>VAL) Alors
            A[I+1] ← A[I]
        Fin Si
    Fin Pour

    // VAL est copié à la position du dernier élément déplacé :
    A[I] ← VAL

    // Edition du résultat :
    Pour I=1 A N Faire
        Ecrire (A[I])
    Fin Pour
Fin
```

Exercice 4. 13. Recherche d'une valeur dans un tableau

Problème: Rechercher dans un tableau d'entiers A une valeur VAL entrée au clavier. Afficher la position de VAL si elle se trouve dans le tableau, sinon afficher un message correspondant. La valeur POS qui est utilisée pour mémoriser la position de la valeur dans le tableau, aura la valeur -1 aussi longtemps que VAL n'a pas été trouvée.

Correction :

```
Variables I, N, VAL, POS : Entiers
Debut
    //Saisie des données:
    Ecrire ("Dimension initiale du tableau A : ")
    Lire (N)

    // Création du tableau A sur la base de la valeur de N :
    Tableau A[N] : Entier
```

```
    // Remplissage du tableau :
    Pour I=1 A N Faire
        Ecrire ("Tapez un entier ")
        Lire (A[I])
    Fin Pour

    // Affichage du tableau :
    Pour I=1 A N Faire
        Ecrire (A[I])
    Fin Pour

    // Saisie de l'élément à rechercher :
    Ecrire ("Elément à rechercher : ")
    Lire (VAL)

    // Recherche de la position de la valeur :
    POS ← -1
    Pour I=1 A N Faire
        Si (A[I]=VAL) Alors
            POS ← I
        Fin Pour
    Fin Pour

    // Edition du résultat :
    Si (POS=-1) Alors
        Ecrire ("La valeur recherchée ne se trouve pas dans le tableau")
    Sinon
        Ecrire ("La valeur ", VAL, " se trouve à la position ", POS)
    Fin Si
Fin
```

Exercice 4. 14.

Ecrire un programme qui demande à l'utilisateur de taper 10 entiers qui seront stockés dans un tableau. Le programme doit trier le tableau par ordre croissant et doit afficher le tableau.
Algorithme suggéré :

- On cherche l'indice du plus petit élément parmi les indices de 0 à 9 et on échange cet élément avec t[0].
- On cherche l'indice du plus petit élément parmi les indices de 1 à 9 et on échange cet élément avec t[1].
- On cherche l'indice du plus petit élément parmi les indices de 2 à 9 et on échange cet élément avec t[2].
- ...
- On cherche l'indice du plus petit élément parmi les indices de 8 à 9 et on échange cet élément avec t[8].

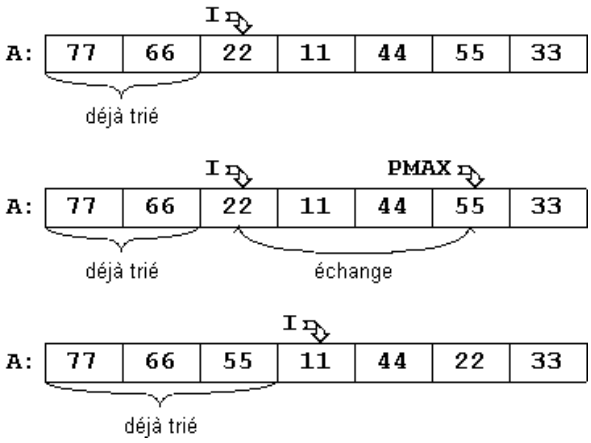
Exercice 4. 15. Fusion de deux tableaux triés

Problème: On dispose de deux tableaux A et B (de dimensions respectives N et M), triés par ordre croissant. Fusionner les éléments de A et B dans un troisième tableau FUS trié par ordre croissant.

Problème : Classer les éléments d'un tableau A par ordre décroissant.

Méthode : Parcourir le tableau de gauche à droite à l'aide de l'indice I. Pour chaque élément $A[I]$ du tableau, déterminer la position $PMAX$ du (premier) maximum à droite de $A[I]$ et échanger $A[I]$ et $A[PMAX]$.

Exemple :



Correction :

```
Variables IA, IB, IFUS, N, M : Entiers
Début
    //Saisie des données: création des tableaux A, B et FUS
    Ecrire ("Dimension du tableau A (max.50) : ")
    Lire (N)

    // Création du tableau A sur la base de la valeur de N :
    Tableau A[N] : Entier

    Ecrire ("Dimension du tableau B (max.50) : ")
    Lire (M)

    // Création du tableau B sur la base de la valeur de M :
    Tableau B[M] : Entier

    // Création du tableau FUS sur la base de la valeur de N et M :
    Tableau FUS[N+M] : Entier

    // Remplissage des tableaux A et B :
    Pour IA=1 A N Faire
        Ecrire ("Tapez un entier ")
        Lire (A[IA])
    Fin Pour
    Pour IB=1 A N Faire
        Ecrire ("Tapez un entier ")
        Lire (B[IB])
    Fin Pour

    // Affichage des tableaux A et B :
    Ecrire ("Tableau A : \n ")
    Pour IA=1 A N Faire
        Ecrire (A[IA])
    Fin Pour
```

```
    Ecrire ("Tableau B : \n")
    Pour IB=1 A N Faire
        Ecrire (B[IB])
    Fin Pour

    /*
    Fusion des éléments de A et B dans FUS de façon à ce que FUS soit aussi
    trié. :
    */
    IA ← 0
    IB ← 0
    IFUS ← 0
    Tant que ((IA<N) ET (IB<M))
        Si (A[IA]<B[IB])
            FUS[IFUS] ← A[IA]
            IFUS++;
            IA++
        Sinon
            FUS[IFUS] ← B[IB]
            IFUS++
            IB++
        Fin Si
    Fin Tant que}

    /*
    Si IA ou IB sont arrivés à la fin de leur tableau, alors copier le
    reste de l'autre tableau.
    */
    Tant que (IA<N)
        FUS[IFUS] ← A[IA]
        IFUS++
        IA++
    Fin Tant que
    Tant que (IB<M)
        FUS[IFUS] ← B[IB]
        IFUS++
        IB++
    Fin Tant que

    // Edition du résultat :
    Ecrire ("Tableau FUS :\n")
    Pour IFUS=1 A N+M Faire
        Ecrire (FUS[IFUS])
    Fin Pour
Fin
```

Exercice 4. 16. Triangle de Pascal

Ecrire un programme qui construit le triangle de $PASCAL$ de degré N et le mémorise dans une matrice carrée P de dimension $N + 1$.

Exemple : Triangle de Pascal de degré 6:

$$N = 0 \qquad 1$$

$N = 1$ 1 1

$N = 2$ 1 2 1

$N = 3$ 1 3 3 1

$N = 4$ 1 4 6 4 1

$N = 5$ 1 5 10 10 5 1

$N = 6$ 1 6 15 20 15 6 1

Méthode:

- Calculer et afficher seulement les valeurs jusqu'à la diagonale principale (incluse). Limiter le degré à entrer par l'utilisateur à 13.
- Construire le triangle ligne par ligne :
  - Initialiser le premier élément et l'élément de la diagonale à 1.
  - Calculer les valeurs entre les éléments initialisés de gauche à droite en utilisant la relation :

$$P_{ij} = P_{(i-1)j} + P_{(i-1)(j-1)}$$

Correction :

```
Variables I, J, N : Entiers
Tableau P[14][14] : Entier

Début
    //Saisie des données:
    Répéter
        Ecrire ("Entrez le degré N du triangle (max.13) : ")
        Lire (N)
    Jusqu'à (N>13 OU N<0);

    /*
    Construction des lignes 0 à N du triangle: Calcul des composantes du
    triangle jusqu'à la diagonale principale.
    */
    Pour I=0 A N Faire
        P[I][I] ← 1
        P[I][0] ← 1;
        Pour J=1 A I-1 Faire
            P[I][J] = P[I-1][J] + P[I-1][J-1]
        Fin Pour
    Fin Pour

    // Edition du résultat
    Ecrire ("Triangle de Pascal de degré : ", N)
    Pour I=0 A N Faire
        Ecrire ("N = ", I)
        Pour J=0 A I Faire
            Si (P[I][J]) Alors
                Ecrire (P[I][J], "\t")    // "\t" ≡ tabulation
            Fin Si
```

```
        Fin Pour
        Ecrire ("\n")                    // "\n" ≡ retour à la ligne
    Fin Pour

Fin
```

Série 5

LES FONCTIONS

Exercice 5. 1.

Ecrire une fonction distance ayant comme paramètres 4 doubles xa, ya, xb *et* yb qui représentent les coordonnées de deux points A et B et qui renvoie la distance AB. Tester cette fonction.

Exercice 5. 2.

Ecrire une fonction f ayant en paramètre un entier et qui renvoie un booléen : OUI si l'entier est premier NON sinon. Tester cette fonction.

Exercice 5. 3.

Ecrire une fonction *swap* ayant en paramètres 2 entiers a et b et qui échange les contenus de a et de b. Tester cette fonction.

Exercice 5. 4.

Etablir une fonction $résoudre$ permettant de calculer les solutions de l'équation :

$$(\boldsymbol{E}) : \ \boldsymbol{ax^2 + bx + c = 0} \qquad \boldsymbol{a, b}\ et\ \boldsymbol{c}\ \text{sont des réels}$$

Cette fonction aura comme paramètres $\boldsymbol{a}$, $\boldsymbol{b}$ et $\boldsymbol{c}$**.** On prendra garde à bien tester tous les cas possibles :

- a est nul, et l'équation est en fait une équation du premier degré. Exemple : $4x - 2 = 0$ donne une unique solution $x = 0.5$.
- Le discriminant $\Delta = (b^2 - 4 * a * c)$ est nul, et il n'y a qu'une seule solution, appelée racine double, au problème. Exemple : $2x^2 + 4x + 2 = 0$ donne $x = -1$.
- Le discriminant Δ est positif, et deux solutions existent : $x_1 = \frac{-b-\sqrt{\Delta}}{2*a}$ et $x_2 = \frac{-b+\sqrt{\Delta}}{2*a}$. Exemple : $2x^2 + x - 6 = 0$ donne $x_1 = 1.5$ et $x_2 = -2$.
- Le discriminant Δ est négatif, et il n'existe pas de solutions (réelles) au problème.