Atelier 01 : Instructions de base d'un algorithme

Exercice 01 : Moyenne de deux nombres

Ecrire un algorithme permettant de calculer la moyenne de deux entiers.

Solution

Algorithme :

```
Algorithme Moyenne
// Moy : est la moyenne de A et B
Variables A, B : entiers
          Moy : réel
Début
   // Saisie des données
   Ecrire ("Entrez la valeur de A:")
   Lire (A)
   Ecrire ("Entrez la valeur de B:")
   Lire (B)
   // Calcul de la moyenne
   Moy ← (A + B) / 2
   // Affichage du résultat
   Ecrire ("La moyenne de A et B est : ", Moy)
Fin
```

Remarque :

Les lignes précédées de deux slashes (//) correspondent à des commentaires en algorithmique.

Exercice 02 : Calcul

Une grande surface accorde à tous ses clients, une réduction de 3% sur

le montant d'achat. Ecrire un algorithme permettant de saisir le montant d'achat (MA) et de calculer le montant de la remise (R) ainsi que le montant à payer (MP).

Solution :

Algorithme :

```
Algorithme Remise
Variables   MA, R, MP : réel
Début
   Ecrire ("Entrez le montant d'achat :")
   Lire (MA)
   R ← MA * 3/100
   MP ← MA – R
   Ecrire ("Le montant de la remise est ", R, " Dh")
   Ecrire ("Le montant à payer est ", MP, "Dh")
Fin
```

Exercice 03 : Permutation de deux variables

Écrire un algorithme qui permute les valeurs de deux variables lues au clavier.

Solution :

Algorithme :

```
Algorithme Permutation
Variables   N1, N2 : réel
                Z : réel
// Z : variable temporaire qui sera utilisée pour permuter N1 et N2
Début
   Ecrire ("Entrez les valeurs de N1 et N2 :")
   Lire (N1, N2)
   Z ← N1
   N1 ← N2
   N2 ← Z
   Ecrire ("Les valeurs de N1 et N2 après permutation sont : N1 = " ,
   N1, " et N2 = " , N2)
Fin
```

Exercice 04 : Evaluation des expressions

Donner les valeurs des variables a, b, c, d, e, f, g, h, i, j et K.

a ← 7 / 2
b ← 7 Div 2
c ← 7 Mod 2
d ← 't' < 'w'
e ← "Maman" > "Papa"
f ← (5 ≠ 2)
g ← "maman" > "Papa"
h ← Non(5=2)
i ← (4<6) et (9>2)
j ← (2 < 0) ou (4 <> 4)
k ← 'A' < 'a'

Solution :

Expression	Valeur de la variable

a ← 7 / 2	3.5
b ← 7 Div 2	3
c ← 7 Mod 2	1
d ← 't' < 'w'	Vrai
e ← "Maman" > "Papa"	Faux
f ← (5 ≠ 2)	Vrai
g ← "maman" > "Papa"	Vrai
h ← Non(5=2)	Vrai
i ← (4<6) et (9>2)	Vrai
j ← (2 < 0) ou (4 <> 4)	Faux
k ← 'A' < 'a'	Vrai

Exercice 05 : Equivalent d'une expression

Donner l'équivalent des expressions booléennes suivantes en utilisant uniquement les opérateurs de comparaison et les opérateurs logiques (Et, Ou et Non).

- x = 2
- x < 6
- x - 2 > 7
- 0 < x < 3
- 3 * x > 18
- a OUex b
- Non (x >= 0)
- Non (Note < 0 Ou Note > 20)

Solution :

Expression	Equivalence

x = 2	Non(x ≠ 2)
x < 6	Non(x > = 6)
x - 2 > 7	Non (x - 2 < = 7)
0 < x < 3	0 <x Et x <3
3 * x > 18	Non(3 * x < = 18)
a OUex b	(a Et Non b) Ou (Non a Et b)
Non (x > = 0)	x < 0
Non (Note < 0 Ou Note > 20)	Non (Note < 0) Et Non(Note > 20)

Exercice 06 : Lois de Morgan

En utilisant des tables de vérité, démontrez les lois suivantes, appelées lois de Morgan.
a = Non(Non(a))
Non(a Et b) = Non(a) Ou Non(b)
Non(a Ou b) = Non(a) Et Non(b)

Solution :

a	**Non (a)**	**Non(Non(a))**
0	1	0
1	0	1

- Donc, il est clair que : a = Non(Non(a))
- Vous faites la même démonstration pour les autres.

Remarque :

Le domaine (Faux, Vrai) est équivalent au domaine (0,1) :
- la valeur 0 corresponde à la valeur booléenne fausse (Faux).
- la valeur 1 corresponde à la valeur booléenne vraie (Vrai).

Exercice 07 : Priorité des opérateurs

Le tableau suivant montre la priorité des opérateurs :

Priorité	**opérateur**

↑ Priorité croissante	( )
	Non
	* / Div Mod
	+ -
	< <= > >=
	= ±
	Et
	Ou
	←

En se basant sur le tableau ci-dessous, donner les valeurs des variables a, b et c.

a ← 4*2 + 5

b ← 5 + 3*2 – 6

c ← a > b Et 7 ± 2 Ou a < b

d ← a > b Et 7 = 2 Ou a < b

Solution :

a =13 , b =5, c = Vrai et d = Faux.

Note :

Les ateliers pratiques de l'ouvrage intitulé :
« Apprendre à programmer : algorithmique et programmation VB (CD-ROM inclus) »
Auteurs : O. El Kharki, J. Mechbouh & D. Ducrot
traduisent tous les algorithmes de cet atelier et ateliers qui suivent en Visual Basic (VB).

Atelier 02 : Les structures alternatives et répétitives

Exercice 01 : Parité

Ecrire un algorithme qui détermine si un nombre entier n saisi au clavier est pair ou impair.

Solution :

On dit qu'un nombre entier n est pair si le reste r de la division entière de n par 2 est égale à 0. Sinon il est impair.

Algorithme :

```
Algorithme Parité
Variables n, r : Entier
Début
   Ecrire ("Entrez la valeur de n : ")
   Lire (n)
   // r : est le reste de la division entière de n par 2
   r ← n mod 2
   Si r = 0 Alors
      Ecrire (n, " est pair")
   Sinon
      Ecrire (n, " est impair")
   FinSi
Fin
```

Rappel : Présentation de l'algorithme

Certaines parties de l'algorithme sont en retrait par rapport à d'autres, c'est ce qu'on appelle ***l'indentation***. Celle-ci est très importante pour la lisibilité de l'algorithme. Elle montre rapidement le début et la fin de chaque instruction ***alternative*** ou ***répétitive*** ainsi le début et la fin

de l'algorithme.
De plus, pour faciliter la compréhension, toutes vos instructions doivent être commentées. Un développeur est souvent amené à modifier un l'algorithme, par conséquent, des commentaires de qualité rendent cette tâche plus facile et plus rapide.

Exercice 02 : Maximum de deux nombres

Ecrire un algorithme qui permet d'afficher le maximum parmi deux nombres saisis au clavier.

Solution :

Algorithme :

Voir le chapitre 02.

Exercice 03 : Condition composée

Ecrire un algorithme qui teste si une note saisie au clavier est comprise entre 0 et 20.

Solution :

Algorithme :

```
Algorithme Tester_une_note
Variable Note : reel
         Message : chaîne
Début
  Ecrire("Donnez une note : ")
  Lire(Note)
  Si Note >= 0 ET Note <= 20 alors
    Message ← "La note " & Note & " est correcte"
  Sinon
    Message ← "La note " & Note & " est incorrecte"
  FinSi
  Ecrire(Message)
Fin
```

Exercice 04 : Nombre de racines d'un trinôme

Ecrire un algorithme permettant de déterminer le nombre de racines réelles d'un trinôme : $a*x^2 + b*x + c = 0$, on suppose que « a » est différent de 0.

Solution :

Soit $\Delta = b^2 - 4ac$. Δ (delta) est appelé le discriminant de ce trinôme.
Si $\Delta = 0$ alors l'équation a une racine réelle double
Si $\Delta > 0$ alors l'équation a deux solutions réelles
Sinon il n'existe aucune racine réelle au trinôme.

Algorithme :

```
Algorithme Nombre_de_racines
Variables a, b, c, delta : Réel
Début
   // Saisie des données
   Ecrire ("Entrez la valeur de a:")
   Lire (a)
   Ecrire ("Entrez la valeur de b:")
   Lire (b)
   Ecrire ("Entrez la valeur de c:")
   Lire (c)
   // Calcul du discriminant
   delta ← b * b  -  4 * a * c
   // Etude du signe du discriminant
   Si delta > 0 alors
      Ecrire (" Deux racines")
   Sinon
      Si delta = 0 alors
         Ecrire ("Une racine double")
      Sinon
         Ecrire ("Pas de racine")
      FinSi
   FinSi
Fin
```

Exercice 05 : Equation du premier degré

Ecrire un algorithme qui résout une équation du premier degré :
A * x + B = 0.

Solution :

La solution de l'équation : A* x + B = 0 dépend du valeurs de A et B.
- si A = 0 et B = 0 alors la solution est l'ensemble des réels
- si A = 0 et B ≠ 0 alors pas de solution
- si A ≠ 0 alors x = -B/A

Algorithme :

```
Algorithme équation_1
Variable A, B, x : réel
Début
   Ecrire("Donner A et B")
   Lire(A, B)
   Si A= 0 alors
      Si B = 0 alors
         Ecrire("Solution : l'ensemble des réels ")
      Sinon
         Ecrire("Pas de solution ")
      FinSI
   Sinon
      x ← - B/A
      Ecrire("La solution est x = ",  x)
   FinSi
Fin
```

Exercice 06 : Simulation d'une calculatrice

Ecrire un algorithme qui permet de saisir deux variables réelles a et b et un opérateur simple : +, -, *, / et d'afficher le résultat.

Solution :

Algorithme :

```
Algorithme  Calculatrice
Variables   A, B, R : Réel
                  OP : caractère
Début
   // Saisie des données
   Ecrire ("Entrez la valeur de A et B : ")
   Lire (A, B)
   // Saisie de l'opérateur
   Ecrire ("Entrez l'opérateur de votre choix : ")
   Lire (OP)
   Selon OP faire
      ' + ' : R ← A + B
      ' – ' : R ← A – B
      ' * ' : R ← A * B
      ' / ' : Si B=0 alors
                 Ecrire ("Division par zéro")
                 // Sortir du programme
                 Quitter
              Sinon
                 R ← A / B
              FinSi
   Sinon
      Ecrire (" Erreur de saisie")
      // Quitter la procédure
      Quitter
   FinSelon
   Ecrire (A, OP, B " = ", R)
Fin
```

Exercice 07 : Mention

Ecrire un algorithme qui lit la moyenne générale (MG) d'un étudiant et affiche la mention.

Solution :

Algorithme :

```
Algorithme Mention
// MG : Moyenne Générale
Variable MG : réel
          Mention : chaîne
Début
   Ecrire("Donner la moyenne générale de l'étudiant : ")
   Lire(MG)
   Si MG >= 16 alors
      Mention ← "Très bien"
   SinonSi MG >= 14 alors
      Mention ← "Bien"
   SinonSi MG >= 12 alors
      Mention ← "Assez bien"
   SinonSi MG >= 10 alors
      Mention ← "Passable"
   Sinon
      Mention ← "Mauvais résultat"
   FinSi
   Ecrire("La mention de l'étudiant est : ", Mention)
Fin
```

Exercice 08 : Maximum de dix nombres

Ecrire un algorithme qui permet d'afficher le maximum parmi dix nombres saisis au clavier.

Solution :

Algorithme :

```
Algorithme  Max
Variables  N, I, Max : Entier
Début
   Max ← 0
   Pour I ← 1 jusqu'à 10 faire
      Ecrire ("Entrer un nombre :")
      Lire (N)
      Si I = 1 ou N > Max Alors
         Max ← N
      FinSi
   FinPour
   Ecrire ("Le nombre le plus grand est : ", Max)
Fin
```

Exercice 09 : Factorielle d'un nombre

Ecrire un algorithme permettant de saisir un nombre entier n et de calculer sa factorielle. On suppose que n > = 1.

Rappel :

n ! = n * (n-1) * (n-2) * (n-3) * ... * 1
n ! = n * (n – 1) !
1 ! = 1
0 ! = 1

Exemple :

5 ! = 5 * 4 * 3 * 2 * 1 = 1 * 2 * 3 * 4 * 5 = 120.

Solution (solution itérative) :

Algorithme :

```
Algorithme  Factorielle
Variables  N, I, F : Entier
Début
   Ecrire ("Entrer un nombre, N> = 2 ")
   Lire (N)
   F ← 1
   Pour I ← N jusqu'à 1 pas -1 faire
      F ← F * I
   FinPour
   Ecrire ("La factorielle est : ", F)
Fin
```

Exercice 10 : Moyenne des notes

Ecrire un algorithme permettant de saisir N notes, de calculer leur somme et leur moyenne de ces notes.

Solution :

Algorithme :

```
Algorithme  Moyenne
Variables  N, Note, S, Moy : Réel
           I : Entier
Début
   S ← 0
   I ← 1
   Ecrire(“Donner le nombre de notes N : ”)
   Lire(N)
   Pour I ← 1 jusqu’à N faire
      Ecrire ("Entrer une note")
      Lire (Note)
      S ← S + Note
   FinPour
   Moy ← S / N
   Ecrire("La somme des notes est : ", S)
   Ecrire("La moyenne des notes est : ", Moy)
Fin
```

Exercice 11 : Rectangle d’étoiles

Ecrire un algorithme qui affiche à l’écran le rectangle d’étoiles ci-dessous. Chaque ligne contient 19 étoiles. Le nombre de lignes est 10.

```
*******************
*******************
*******************
*******************
*******************
*******************
*******************
*******************
*******************
*******************
```

Solution :

Algorithme :

```
Algorithme Rectangle_des_étoiles
// I : compteur, J : compteur
Variables I, J : entiers
Début
   Pour I ← 1 jusqu'à 10 faire
      Pour J ← 1 jusqu'à 19 faire
         Ecrire("*")
      FinPour
      // Retour à la ligne
      Ecrire("\n")
   FinPour
Fin
```

Remarque :

L'instruction ***Ecrire("\n")*** est le retour à la ligne en algorithmique.

Exercice 12 : triangle d'étoiles

Ecrire un algorithme qui affiche à l'écran le triangle d'étoiles suivant :

```
*
**
***
****
*****
```

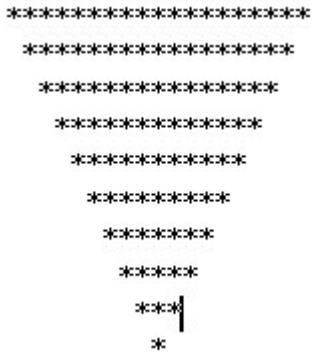
Solution :

Algorithme :

```
Algorithme Triangle_des_étoiles
// i : compteur, j : compteur
Variables I, J : entiers
Début
   Pour I ← 1 jusqu'à 5 faire
      Pour J ← 1 jusqu'à I faire
         Ecrire("*")
      FinPour
      // Retour à la ligne
      Ecrire("\n")
   FinPour
Fin
```

Exercice 13 : Pyramide d'étoiles

Ecrire un algorithme qui affiche à l'écran la pyramide d'étoiles ci-dessous. Chaque ligne comporte 19 caractères (espaces ou étoiles (*)).



Solution :

Chaque ligne comporte 19 caractères (espaces ou étoiles (*)). Soit :
n_etoiles : nombre d'étoiles dans une ligne
n_espaces : nombre d'espaces dans une ligne
Donc une ligne comporte (n_espaces /2) et n_etoiles puis (n_espaces /2).

Algorithme :

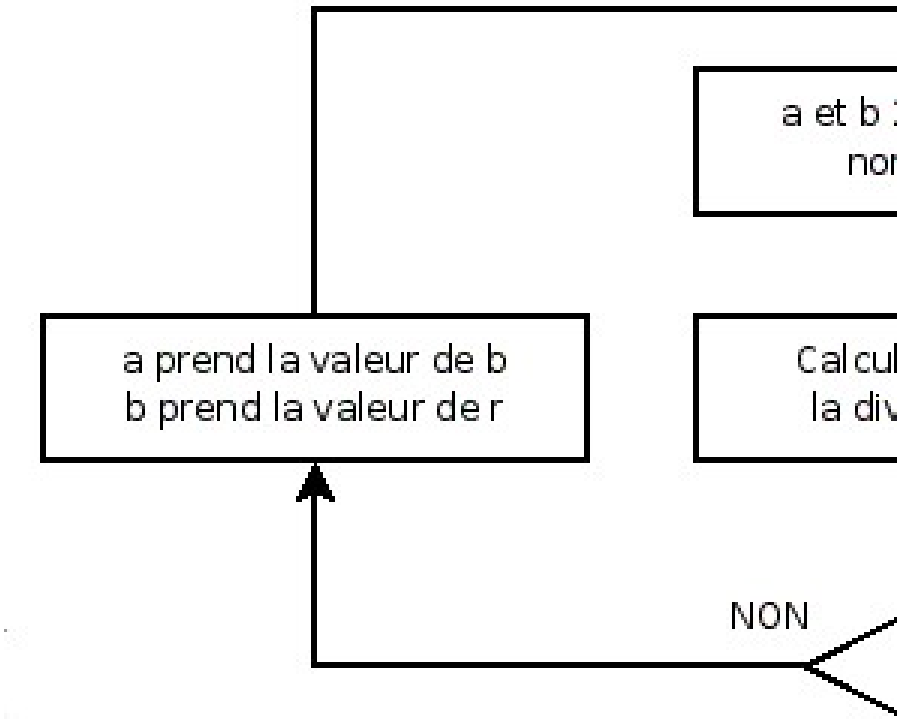
```
Algorithme  pyramide_inversée
// n_etoiles : nombre d'étoiles dans une ligne
// n_espaces : nombre d'espaces dans une ligne
Variables  n_etoiles, n_espaces, i, j, k : Entier
Début
   n_etoiles = 19
   Tant que n_etoiles  >= 1 faire
      n_espaces = 19 -  n_etoiles
      i  = 1
      Tant que i <= (n_espaces /2) faire
         écrire (" ")
         i = i + 1
      FinTantQue
      j = 1
      Tant que j < n_etoiles  faire
         écrire ("*")
         j = j +1
      FinTantQue
      k = 1
      Tant que k <= (n_espaces /2) faire
         écrire (" ")
         k = k+1
      FinTantQue
      n_etoiles  = n_etoiles  - 2
      écrire ("\n ")
   FinTantQue
Fin
```

Exercice 14 : PGDC

Ecrire un algorithme qui calcul le PGDC (plus grand diviseur commun) de deux nombre entier a et b non nuls (a > b).

Solution :

La figure suivante présente le principe de calcul du PGDC de a et b.


a et b :
nor
a prend la valeur de b
b prend la valeur de r
Calcul
la div
NON

Algorithme :

```
Algorithme PGDC
Variable a, b, r : entiers
Début
   Ecrire("Donner a et b")
   Lire(a, b)
   r ← a mod b
   Tant que r ≠0 faire
      a ← b
      b ← r
      r ← a mod b
   FinTantQue
   Ecrire(" Le PGDC est : ", b)
Fin
```

Exercice 15 : Suite numérique

Soit la suite numérique U_n suivante :

$$\begin{cases} \text{Si } n = 0, U_0 = 4 \\ \text{Si } n > 0, U_n = 5\,U_{n-1} + 9 \end{cases}$$

Ecrire un algorithme qui calcul le terme U_n et la somme Sn. ($Sn = U_0 + U_1 + U_2 + \ldots + U_{n-1} + U_n$).

Solution :

Algorithme :

```
Algorithme Suite_numérique
Variable Un, Sn : réel
         i, n : entier
Début
   Ecrire("Donner n")
   Lire(n)
   // Initialisation : Sn = 4 et Un = 4
   Sn ← 4
   Un ← 4
   Pour i ← 1 jusqu'à n faire
      Un ← 5Un + 9
      Sn = Sn + Un
   FinPour
   Ecrire ("Le terne U", n, " = ", Un)
   // "\n" : retour à la ligne
   Ecrire("\n")
   Ecrire ("La somme S", n, " = ", Sn)
End
```

Exercice 16 : Suite de Fibonacci

Ecrire un algorithme qui détermine le terme U_n de la suite de Fibonacci définit comme suit :

$$\begin{cases} U_0 = 0 \\ U_1 = 1 \\ U_n = U_{n-1} + U_{n-2} \text{ , } n >= 2 \end{cases}$$

Solution :

Algorithme :

```
Algorithme suite_Fibonacci
// Um_1 : est le terme Um-1
// Um_2 : est le terme Um-2
// Um : est le terme Um
Variables Um, Um_1, Um_2 : entier
          n, m : entier
Début
   Ecrire("Donnez la valeur de n, n > = 2")
   Lire(n)
   Um_2 ← 0
   Um_1 ← 1
   m ← 2
   TantQue m <= n faire
      Um ← Um_1 + Um_2
      Um_2 ← Um_1
      Um_1 ← Um
      m ← m +1
   FinTantQue
   Ecrire("Le terme Un = ", Um)
Fin
```

Atelier 03 : Opérations sur les tableaux

Exercice 01 : Nombre de moyennes > = 10

En utilisant les tableaux, écrire un algorithme qui permet la saisie d'une liste de n moyennes réelles et d'afficher le nombre des moyennes supérieures ou égales à 10. On suppose que n < =100

Solution :

Algorithme :

```
Algorithme Moy_sup_à_10
// M : indice supérieur du tableau
Constante M = 100
// Moy : tableau de M éléments réels
Variable Moy : tableau[1..M] des réels
           i, n, k : entiers
Début
   Ecrire("Donner la valeur de n : ")
   Lire(n)
   // Saisie des éléments du tableau
   Pour i ← 1 jusqu'à n Faire
      Ecrire ("Donner l'élément ", i, " du tableau")
      Lire (Moy[i])
   FinPour
   // k : compte le nombre de moyennes supérieures ou égales à 10.
   k ← 0
   Pour i ← 1 jusqu'à n Faire
      Si Moy[i] > = 10 alors
         k ← k + 1
      FinSI
   FinPour
   // Affichage du résultat
   Ecrire("Le nombre de moyennes supérieures ou égales à 10 est ", k)
Fin
```

Remarque :

- On peut ajouter dans l'algorithme les instructions suivantes pour gérer les cas où l'utilisateur saisisse une valeur supérieure à 100 (indice supérieur du tableau) :

```
Si n > 100 alors
   n ← M
FinSI
```

- Remarquer bien l'avantage de la constante M qui peut être modifiée facilement pour la maintenance des programmes.

Exercice 02 : Permutation des éléments d'un tableau

Ecrire un algorithme qui permute les éléments d'un tableau de 8 éléments en plaçant le dernier élément en premier et ainsi de suite, voir la figure suivante :

15	18	13	99	68	71	96	55

Tableau initial

55	96	71	68	99	13	18	15

Tableau après permutation

Solution :

Algorithme :

```
Algorithme Permutation_tableau
Variables T : Tableau[1..8] d'entiers
          I, X, K : Entier
Début
  Pour I ← 1 jusqu'à 8 Faire
    Ecrire ("Donner l'élément ", I, " du tableau")
    Lire (t[I])
  FinPour
  K ← 8
  // Notez bien que I varie de 1 à 4 dans la boucle suivante :
  Pour I ← 1 jusqu'à 4 Faire
    X ← T(K)
    T(K) ← T(I)
    T(I) ← X
    K ← K – 1
  FinPour
  Pour I ← 1 jusqu'à 8 Faire
    Ecrire(T[i])
  FinPour
Fin
```

Exercice 03 : Maximum des moyennes d'une classe

Ecrire un algorithme qui permet la saisie des moyennes d'une classe de n étudiants et affiche le maximum.

Solution :

Algorithme :

```
Algorithme maximum_tableau
// M : indice supérieur du tableau
Constante M = 100
Variable Moy : tableau[1 .. M] de réels
          i, n : entier
          max : réel
Début
   Ecrie("donner le nombre d'étudiants")
   Lire(n)
   Si n > M alors
      Ecrire("Dépassement de capacité du tableau")
      n ← M
   FinSI
   Pour I ← i jusqu'à n faire
   Ecrire ("Entrer la moyenne ",i, " :")
      Lire(Moy[i])
   FinPour
   max ← Moy[1]
   Pour i ← 2 jusqu'à n faire
      Si  Moy[i] > max alors
         Max ← Moy[i]
      FinSI
   FinPour
   Ecrire ("le maximum des moyennes est ", max)
Fin
```

Exercice 04 : Trier un tableau

Ecrire un algorithme qui permet la saisie de 4 moyennes et de les

classer selon l'ordre croissant.

Solution :

Algorithme :

```
Algorithme Trier_tableau
// N : indice supérieur du tableau
Constante N = 4
Variables Moy : tableau[1 .. N] de réels
          I, J : entier
          X : réel
Début
  // Saisie des éléments du tableau
  Pour I ← 1 jusqu'à N faire
    Ecrire ("Entrer la moyenne ",I, " :")
    Lire(Moy[I])
  FinPour
  // Tri des éléments du tableau
  Pour I ← 1 jusqu'à N-1 faire
    Pour J ← 1 +1 jusqu'à N faire
      Si  Moy[I] > Moy[J]  alors
        X ← Moy[I]
        Moy[I] ← Moy[J]
        Moy[J] ← X
      FinSI
    FinPour
  FinPour
  // Affichage des éléments du tableau trié
  Pour I ← 1 jusqu'à N faire
    Ecrire (Moy[I])
  FinPour
Fin
```

Exercice 05 : Tableau à deux dimensions

Ecrire un algorithme qui permet :

- la saisie des notes d'une classe de 5 étudiants en 4 matières
- calcul et affiche la moyenne de chaque étudiant
- calcul et affiche la moyenne de la classe dans chaque matière
- calcul et affiche la moyenne générale de la classe.

Solution :

Algorithme :

```
Algorithme Tableau_2_dimensions
// N : nombre d'étudiants, M : nombre de matières
Constante N = 5, M = 4
// Notes[I,J] : note de l'étudiant I dans la matière J
// Moy_E[I]  : Moyenne de l'étudiant I
// Moy_M[J]  : Moyenne de la classe dans la matière J
// Moy_G : Moyenne générale de la classe
Variable Notes : tableau[1..N, 1..M] des réels
         Moy_E : tableau[1..N] des réels
         Moy_M : tableau[1..M] des réels
          I, J : entiers
          S, Moy_G : réels
Début
  // Saisie des notes
  Pour I ← 1 jusqu'à N Faire
    Pour J ← 1 jusqu'à M Faire
      Ecrire ("Donner la note de l'étudiant ", I, " dans la matière", J)
      Lire (Notes[I, J])
    FinPour
  FinPour
  // Calcul et affichage de la moyenne de chaque étudiant
  Pour I ← 1 jusqu'à N Faire
    S ← 0
    Pour J ← 1 jusqu'à M Faire
      S ← S + Notes[I, J]
    FinPour
    Moy_E[I] ← S / M
  FinPour
  Pour I ← 1 jusqu'à N Faire
    Ecrire("La moyenne de l'étudiant ", I, " est ",  Moy_E[I])
  FinPour
  // Calcul et affichage de la moyenne par matière
  Pour J ← 1 jusqu'à M Faire
    S ← 0
    Pour I ← 1 jusqu'à N Faire
      S ← S + Notes[I, J]

    Moy_M[J] ← S / N
  FinPour
```

Exercice 06 : Produit de deux matrices

Ecrire un algorithme qui calcule la matrice C qui est le produit de deux matrices A et B. A est de l'ordre n*m et B est de l'ordre m*k.

Solution :

Exemple :

A				B			C	
1	1	1		1	1	=	3	3
1	1	1	**X**	1	1		3	3
				1	1			

A est de l'ordre 2 * 3, B est de l'ordre 3*2 et C est de l'ordre 2*2. la valeur d'un élément de la matrice C est définie comme suit :

$$C(i,j) = \sum_{r=1}^{r=m} A(i,r) * B(r,j)$$

Dans cet exemple m = 3.

Note :
Le produit de deux matrices exige que le nombre de colonnes de la première matrice soit égal au nombre de lignes de la deuxième matrice.

Algorithme :

```
Algorithme Produit_deux_matrices
Variable A: tableau [1 To 20, 1 To 20] de réels
         B: tableau [1 To 20, 1 To 20] de réels
         C: tableau [1 To 20, 1 To 20] de reels
         i, j, r, n, m, k : entier
début
  Ecrire("Donnez le nombre de lignes de la matrice A : ")
  lire(n)
  Ecrire("Donnez le nombre de colonnes de la matrice A : ")
  lire(m)
  Ecrire("Donnez le nombre de colonnes de la matrice B : ")
  lire(k)
  //Remplissage de la matrice A
  Pour i ← 1 jusqu'à n faire
    Pour r ← 1 jusqu'à m faire
      Ecrire("Donnez l'élément A[" , i , "," , r,"]"))
      Lire(A[i, r])
    FinPour
  FinPour
  //Remplissage de la matrice B
  Pour r ← 1 jusqu'à m faire
    Pour j ← 1 jusqu'à k faire
      Ecrire("donnez l'élément B[",  r , "," , j , "]"))
      Lire(B(r, j))
    FinPour
  FinPour
  //Calcul de la matrice produit C
  Pour i ← 1 jusqu'à n faire
    Pour j ← 1 jusqu'à k faire
      C[i, j] = 0
      Pour r ← 1 jusqu'à m faire
        C[i, j] = C[i, j] + A[i, r] * B[r, j]
      FinPour
    FinPour
  FinPour
  //Affichage des éléments de la matrice produit C
  Pour i ← 1 jusqu'à n faire
    Pour  j ← 1 jusqu'à k faire
      Ecrire( "C[" , i ,  "," ,j ,") = ", C(i, j))
```

Exercice 07 : Tableau dynamique à deux dimensions

En se basent sur l'annexe 01 qui traduit les instructions algorithmique en Visual Basic (voir la fin de l'ouvrage), écrire un programme VB qui utilise un tableau dynamique à deux dimensions (matrice) puis redimensionner le tableau en demandant le nombre de lignes et de colonnes. Ajouter des instructions pour remplir et afficher le contenu de ce tableau.

Solution :

Programme VB :

```
Sub main()
   Dim A() As Double
   Dim i As Integer, j As Integer
   Dim n As Integer, m As Integer
   n = Val(InputBox("Donnez le nombre de lignes de la matrice A"))
   m = Val(InputBox("Donnez le nombre de colonnes de la matrice A"))
   ' Redimensionner la matrice A
   ReDim A(n, m)
   ' Remplissage de la matrice A
   For i = 1 To n
      For j = 1 To m
         A(i, j) = Val(InputBox("Donnez l'élément A(" & i & "," & j &
         ")"))
      Next j
   Next i
      ' Affichage des éléments de la matrice A
   For i = 1 To n
      For j = 1 To m
         MsgBox "A(" & i & "," & j & ")= " & A(i, j)
      Next j
   Next i
End Sub
```

Exercice 08 : Recherche d'une valeur dans un

tableau

Ecrire un algorithme qui cherche une valeur x dans un tableau t.

Solution :

Algorithme :

```
Algorithme recherche
Variable t : tableau[100] de réels
         x : réel
         trouvé : logique
         n, i, j : entier
Début
  Ecrire("Donner la taille du tableau")
  Lire(n)
  Si n > 100 alors
    n ← 100
  FinSI
  // Remplissage du tableau
  Pour i ← 1 jusqu'à n faire
    Ecrire("Donner l'élément t[", i, "]")
    Lire(t[i])
  FinPour
  // Saisie da la valeur recherchée
  Ecrire("Donner la valeur recherchée")
  Lire(x)
  // Recherche dans le tableau
  trouvé ← Faux
  Pour i ← 1 jusqu'à n faire
    SI t[i] = x alors
      j ← i
      trouvé ← Vrai
      // sortir de la boucle Pour
      Quitter Pour
    FinSI
  FinPour
  Si trouvé = Vrai alors
    Ecrire(" La valeur ", x, " est trouvée, elle est au rang ", j)
  Sinon
    Ecrire(" La valeur ", x, " n'existe pas ")
  FinSI
Fin
```

Note : Autres exercices sur les tableaux

L'atelier 08 contient d'autres exercices sur les tableaux. Il aborde les algorithmes de tri et de recherche.

Atelier 04 : Les structures

Exercice 01 : Structure personne

Ecrire un algorithme qui :
- définit une structure **Personne** qui contient trois champs : nom, prénom et âge.
- déclare deux variables etudiant1 et etudiant2 de type Personne.
- calcule la différence d'âge entre les deux étudiants.

Solution :

Algorithme :

```
Algorithme différence_age

// Déclaration de la structure Personne
Type Structure Personne
   nom : chaîne
   prénom : chaîne
   age : entier
FinStruct
// Déclaration des variables de type Personne
Variables etudiant1, etudiant2 : Personne

// Algorithme principal
Début
   Ecrire("Entrez le nom, le prénom puis l'âge de l'étudiant 1")
   Lire(etudiant1.nom, etudiant1.prénom, etudiant1.age)
   Ecrire("Entrez le nom, le prénom puis l'âge de l'étudiant 2")
   Lire(etudiant2.nom, etudiant2.prénom, etudiant2.age)
   Ecrire("La différence d'âge entre ", etudiant1.nom, " et ",
   etudiant2.nom, " est de ")
   Si etudiant1.age > etudiant2.age Alors
      Ecrire(etudiant1.age – eudiant2.age, " ans")
   Sinon
      Ecrire(etudiant2.age – etudiant1.age, " ans")
   FinSi
Fin
```

Exercice 02 : Structure imbriquée

Etudiant est une structure composée de trois champs : nom, prénom et date_de_naissance. Nom et prénom sont de type chaîne. Date_de_naissance est de type **N_date**.
N_date est une structure composée de trois champs : jour, mois et année. Jour et année sont de type entiers, mois est de type chaîne.
Ecrire un algorithme qui permet de saisir et d'afficher l'année de naissance d'un étudiant.

Solution :

Algorithme :

```
Algorithme Année_de_naissance

// Déclaration de la structure N_Date
Type Structure N_Date
   jour : entier
   mois : chaîne
   année : entier
FinStruct

// Déclaration de la structure Etudiant
Type Structure Etudiant
   nom : chaîne
   prénom : chaîne
   date_naissance : N_date
FinStruct
// Déclaration de la variable etudiant1 de type Etudiant
Variable etudiant1 : Etudiant

// Algorithme principal
Début
   Ecrire("Entrer l'année de naissance de l'etudiant1")
   Lire(etudiant1.date_naissance.année)
   Ecrire("L'année de naissance de l'etudiant1 est : ",
   etudiant1.date_naissance.année)
Fin
```

Atelier 05 : Les fonctions prédéfinies

Exercice 01 : Nombre de mots dans une phrase

Ecrire un algorithme permettant de saisir une chaîne de caractère (une phrase) et d'afficher le nombre de mots de cette chaîne.

Solution :

La solution consiste à compter le nombre d'espaces dans la chaîne de caractères. Le nombre de mots est égal au nombre d'espaces + 1.

Algorithme :

```
Algorithme  nombre_de_mots
Variables phrase : chaîne
          N, I, Compteur : Entier
Début
   Ecrire("Entrer une chaîne de caractères :")
   Lire (phrase)
   N ← longueur(phrase)
   Compteur ← 0
   Pour I ← 1 jusqu'à N Faire
      Si copie(phrase, I, 1) = " " alors
         Compteur ← Compteur + 1
      FinSi
   FinPour
   Ecrire("Cette chaîne contient ", Compteur + 1, " mots")
Fin
```

Exercice 02 : Nombre de voyelles dans une phrase

Ecrire un algorithme permettant de saisir une phrase et d'afficher le nombre de voyelles contenu dans cette phrase en utilisant la fonction de recherche.

Solution :

La solution consiste à stocker toutes les voyelles dans la variable ***V*** de type chaîne de caractères. Grâce à la fonction ***Recherche***, on examine chaque caractère de la chaîne ***Ch*** saisie au clavier s'il existe dans la chaîne des voyelles ou non.

Exemple :

Soit :
V = "aeiouy"
Ch = "Je suis un étudiant".
La chaîne Ch contient 7 voyelles.

Algorithme :

```
Algorithme  voyelle
Variables Ch,V : chaîne
          N, I , Compteur : Entier
Début
  Ecrire("Entrer une chaîne de caractères :")
  Lire (Ch)
  N ← longueur(Ch)
  Compteur  ← 0
  V ← "aeiouy"
  Pour I ← 1 jusqu'à N Faire
    Si recherche(V, copie(Ch, I, 1)) ≠ 0 alors
      Compteur  ← Compteur  + 1
    FinSi
  FinPour
  Ecrire("Cette chaîne contient ", Compteur , " voyelles")
Fin
```

Exercice 03 : Remplacement d'un caractère par un autre dans une phrase

Soit une phrase qui contient des mots séparés par des slashes « / », voir l'exemple suivant :

"Sabir / Ahmed / étudiant en troisième année / Tél : 023 25 86 89"

Ecrire un algorithme qui remplace les slashes « / » par des point virgules « ; ».

Solution :

Le résultat souhaité est de transformer la phrase suivante :

"Sabir / Ahmed / étudiant en troisième année / Tél : 023 25 86 89"

En :

"Sabir ; Ahmed ; étudiant en troisième année ; Tél : 023 25 86 89"

Algorithme :

```
Algorithme Remplacement_ caractères
Variable phrase1, phrase2 : chaîne
         Z : caractère
         I : entier
Début
  Phrase1 ← "Sabir / Ahmed / étudiant en troisième année / Tél : 023
  25 86 89"
  phrase2 ←  ""
  Pour I ← 1 jusqu'à longueur(phrase1) Faire
    Si Copie(phrase1, I,  1) = "/" alors
      Z ←  ';'
    Sinon
      Z ←  Copie(phrase1, I,  1)
    FinSI
    phrase2 = phrase2 & z
  FinPour
  Ecrire("phrase2")
Fin
```

Exercice 04 : Affichage d'une chaîne en inverse

Ecrire un algorithme permettant de saisir une chaîne de caractère et de l'afficher en inverse.

Solution :

Algorithme :

```
Algorithme  chaine_en _inverse
Variables ch , ch1, temp : chaîne
            N, i : Entier
Début
   Ecrire("Entrer une chaîne de caractères:")
   Lire (ch)
   // N est la longueur de la chaine ch
   N ← longueur(ch)
   Temp ← ""
   Ch1 ← ""
   Pour i ← N jusqu'à 1 pas -1 Faire
         temp ← Copie(ch, i,  1)
         ch1 ← ch1 & temp
   FinPour
   Ecrire(ch1)
Fin
```

Exercice 05 : Concaténation des chaînes de caractères

Écrire un algorithme qui demande à l'utilisateur d'entrer une chaîne de caractères (une date) sous la forme JJMMAA et qui affiche cette date sous la forme JJ/MM/20AA.

Solution :

La date résultat date2 (figure suivante) sera le résultat de la concaténation des deux premiers caractères de date1, suivis du caractère "/", suivis des deux caractères du milieu de date1, suivis des caractères "/20", puis des deux derniers caractères de date1.

Date saisie (date1)						Date résultat (date2)									
0	3	0	9	0	8	0	3	/	0	9	/	2	0	0	8

Algorithme :

```
Algorithme concaténation
Variables date1, date2 : chaîne
Début
   Ecrire("Saisir une date sous la forme JJMMAA : ")
   Lire(date1)
   date2 ← Concat(Copie(date1,1,2), "/", Copie(date1,3,2), "/20",
   Copie(date1,5,2))
   Ecrire(date2)
Fin
```

Atelier 06 : Les fonctions et procédures

Exercice 01 : Fonction Moyenne

En se basant sur l'algorithme, de l'exercice 01 de l'atelier 01, écrire le sous-algorithme de la fonction ***Moyenne*** qui renvoie la moyenne de deux entiers.
Ecrire un algorithme qui contient la déclaration de la fonction moyenne et des instructions qui appellent cette fonction.

Solution :

Sous-algorithme de la fonction Moyenne :

```
// Déclaration de la fonction Moyenne
Fonction Moyenne(X : entier, Y : entier) : réel
Début
   Retourner  (X + Y) / 2
FinFonct
```

Algorithme :

```
Algorithme Fonction_Moyenne
Variables A, B : entiers

// Déclaration de la fonction Moyenne
Fonction Moyenne(X : entier, Y : entier) : réel
Début
   Retourner (X + Y) / 2
FinFonct

// Programme principal
Début
   // Saisie des données
   Ecrire ("Entrez la valeur de A:")
   Lire (A)
   Ecrire ("Entrez la valeur de B:")
   Lire (B)
   // Appel de la fonction Moyenne
   Ecrire ("La moyenne de A et B est : ", Moyenne(A, B))
Fin
```

Exercice 02 : Fonction $f(x) = 3x^3 + 4x + 8$

Écrire un algorithme qui contient la déclaration de fonction $f(x) = 3x^3 + 4x + 8$ et affiche le résultat de cette fonction pour $x = 1$, $x = 2$ et $x = 2,7$.

Solution :

Algorithme :

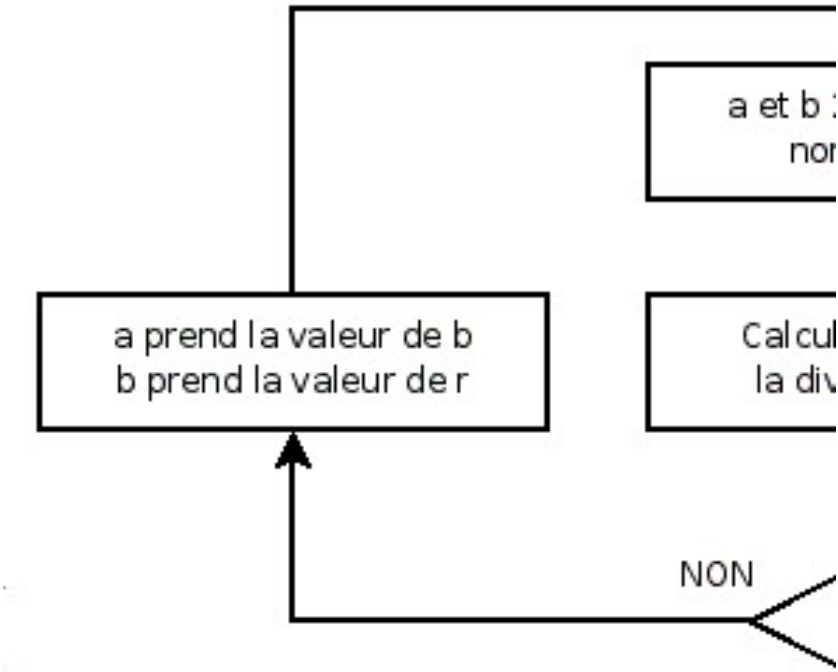
```
Algorithme Fonction_polynomiale

// Déclaration de la fonction f(x) = 3x^3 + 4x + 8
Fonction f(x : réel) : réel
Début
   Retourner 3*x^3 + 4*x + 8
FinFonct

// Programme principal
Début
   //Appel de fonction
   Ecrire("f(1) = ", f(1), " f(2) = ", f(2), " f(2.7) = ", f(2.7))
Fin
```

Exercice 03 : Fonction PGCD récursive

Ecrire un algorithme qui calcul le PGDC (plus grand diviseur commun) de deux nombre entier a et b non nuls (a > b) en utilisant une fonction récursive.

Solution :

La figure suivante présente le principe de calcul du PGDC de a et b.



Exemple :

PGCD(5, 2) = 1 ; PGCD(10, 5) = 5 ; PGCD(12, 8) = 4.

Algorithme :

```
Algorithme PGCD_récursive
Variable a, b : entier

// Déclaration de la fonction récursive qui calcul le PGCD
Fonction PGCD(x : entier, y : entier) : entier
   Variable r : entier
   Si y = 0 alors
      Retourner x
   Sinon
      // r : reste de la division entière de x par y
      r ← x mod y
      Retourner PGCD(y, r)
   FinSi
FinFonction

// Algorithme principal
Début
   Ecrire((" Donnez a et b : "))
   Lire(a, b)
   Ecrire("PGCD(a, b) = ", PGCD(a, b))
Fin
```

Exercice 04 : Fonction somme récursive

Ecrire un algorithme qui calcul :
Somme(m) = 1 + 2 + 3 + 4 + ... + m-1 + m en utilisant une fonction récursive.

Solution :

Algorithme :

```
Algorithme Somme_récursive
Variable m : entier

// Déclaration de la fonction somme récursive
Fonction somme(n : entier) : entier
Début
   Si n = 0 alors
      Retourner 0
   Sinon
      Retourner somme(n - 1) + n
   FinSi
FinFonct

// Programme principal
Début
   Ecrire("Donnez m")
   Lire(m)
   Ecrire("La somme de m premiers entiers est : ", somme(m))
Fin
```

Exercice 05 : Passage des paramètres par adresse

Qu'affiche à l'écran l'algorithme suivant :

```
Algorithme Passage_par_adresse
Variables n, p : entiers

// Déclaration de la procédure echange
Procédure echange(Var a : entier, Var b : entier)
Variable c : entier
Début
   Ecrire("Début echange ", a, " ", b)
   c ← a
   a ← b
   b ← c
   Ecrire("Fin echange ", a, " ", b)

FinProc

// Programme principal
Début
   n ←  10
   p ←  20
   Ecrire("Avant l'appel ", n, " ", p)
   echange n, p
   Ecrire("Fin echange ", n, " ", p)
Fin
```

Solution :

Dans la ligne de code : ***Procédure echange(Var a : entier, Var b : entier)***, il y a la présence du mot clé ***Var***. Il s'agit donc d'un passage de paramètres par adresse.

Ce programme affiche à l'écran :
Avant l'appel 10 20
Début echange 10 20
Fin echange 20 10
Fin echange 20 10

Exercice 06 : Passage des paramètres par valeur

Soit l'algorithme suivant :

```
Algorithme passage_par_valeur
Variables n, p : entiers

// Déclaration de la procédure echange
Procédure echange(a : entier, b: entier)
Variable c : entier
Début
   Ecrire("Début echange ", a, " ", b)
   c ← a
   a ← b
   b ← c
   Ecrire("Fin echange ", a, " ", b)
FinProc

// Programme principal
Début
   n ←  10
   p ←  20
   Ecrire("Avant l'appel ", n, " ", p)
   echange n, p
   Ecrire("Fin echange ", n, " ", p)
Fin
```

- Quelle est la différence entre l'algorithme de l'exercice 05 et celui de l'exercice 06.
- Qu'affiche cet algorithme à l'écran

Solution :

La seule différence est l'absence du mot clé ***Var*** dans l'algorithme de l'exercice 06. Il s'agit d'un passage de paramètres par valeur dans ce dernier.

Ce programme affiche à l'écran :
Avant l'appel 10 20
Début echange 10 20
Fin echange 20 10
Fin echange 10 20

Remarque :

Les valeurs de n et p ne changent pas après l'appel de la procédure ***echange*** car le passage des paramètres est par valeur.

Exercice 07 : Suite numérique

Soit la suite numérique Un suivante :

$$\begin{cases} \text{Si } n = 0, U_0 = 4 \\ \text{Si } n > 0, U_n = 5\ U_{n-1} + 9 \end{cases}$$

Ecrire un algorithme qui calcul le terme U_n en utilisant une fonction récursive.

Solution :

Algorithme :

```
Algorithme suite_numérique
Variable n : entier

// Déclaration de la fonction U qu calcule le terme Um = U(m)
Fonction U(m : entier) : réel
   Si m = 0  alors
      Retourner 4
   Sinon
      Retourner 5 * U(m - 1) + 9
   FinSI
FinFonction

// Algorithme principal
Début
   Ecrire("Donner n")
   Lire(n)
   Ecrire("Le terne U",  n, " = ", U(n)
Fin
```

Exercice 08 : Partie entière

Ecrire un algorithme qui contient une fonction E(x) qui renvoie la

partie entière d'un nombre réel saisi au clavier.

Exemple

E(2) = 2 E(6.2) = 6
E(-4) = -4 E(-10.2) = -11

Solution :

Soit x un nombre réel. Si x >= 0 alors E(x) = x div 1 (la partie entière de x est égal à la division entière de x par 1).
Si x < = 0, il y a deux cas : si x est différent de (x div 1) dans ce cas E(x) = (x div 1) – 1 sinon E(x) = x div 1.

Algorithme :

```
Algorithme Partie_entière
Variable x : réel

// Déclaration de la fonction E : partie entière
Fonction E(y : réel) : entier
   Variable r : entier
   Si y > = 0 alors
      //L'opérateur div : c'est la division entière
      Retourner  y div 1
   Sinon
      r ← y div 1
      Si y ≠ r alors
         Retourner  r - 1
      Sinon
         Retourner y
      FinSI
   FinSi
FinFonct

// Programme principal
Début
   Ecrire("Donnez x : ")
   Lire(x)
    ‘ Appel de la fonction
   Ecrire("E(",  x, ") = ", E(x))
Fin
```

Exercice 09 : Suite de Fibonacci

En utilisant une fonction récursive, écrire un algorithme qui détermine le terme U_n de la suite de Fibonacci définit comme suit :

$$\begin{cases} U_0 = 0 \ ; \ U_1 = 1 \\ U_n = U_{n-1} + U_{n-2} \ , \ n >= 2. \end{cases}$$

Solution :

L'algorithme :

```
Algorithme suite_fibonacci
Variable n : entier

// Déclaration de la fonction fibonacci
Fonction fibonacci(k : entier) : entier
   Si k = 0 alors
      Retourner 0
   SinonSI k = 1 alors
      Retourner 1
   Sinon
      Retourner fibonacci(k - 2) + fibonacci(k - 1)
   FinSi
FinFonction

// Programme principal
Début
   Ecrire("Donnez la valeur de n")
   Lire(n)
   Ecrire("Le terme U", n, " = ", fibonacci(n))
Fin
```

Exercice 10 : Fonction factorielle récursive

Ecrire un algorithme qui calcule la factorielle d'un nombre entier positif en utilisant la récursivité.

Solution :

Algorithme :

```
Algorithme calcul_factorielle
Variable n : entier

// Déclaration de la fonction Fact qui calcule la factorielle d'un nombre
Fonction Fact (m : entier) : entier
Début
   Si m= 0 Alors
      Retourner 0
   Sinon
      Retourner Fact(m-1) *m
   FinSi
FinFonct

// Programme principal
Début
   Ecrire((" Donner un entier n positif : "))
   Lire(n)
   // Appel de la fonction Fact
   Ecrire("La factorielle de n est : ", fact(n))
Fin
```

Exercice 11 : Fonctions et structures

Soit :
- **Personne** une structure qui contient trois champs : nom, prénom et âge.
- **Etudiant1** et **Etudiant2** deux variables de type Personne.
- **Différence_age** est une fonction qui renvoie la différence d'âge entre l'etudiant1 et l'etudaint2.

Ecrire un algorithme qui permet de saisir l'âge de l'etudiant1 et l'etudiant2 et affiche la différence d'âge entre les deux étudiants (utiliser la fonction Différence_age**).**

Solution :

Algorithme :

```
Algorithme Différence_age

// Déclaration de la structure Personne
Type Structure Personne
   nom : chaîne
   prénom : chaîne
   age : entier
FinStruct
// Déclaration de deux variables de type Personne
Variables Etudiant1, Etudiant2 : Personne

// Déclaration de la fonction Différence_age
Fonction Différence_age (Etudiant1, Etudiant2 : Personne) : entier
Variable y : entier
Début
   Si Etudiant1.age > Etudiant2.age Alors
      y ← Etudiant1.age – Etudiant2.age
   Sinon
      y ←  Etudiant2.age – Etudiant1.age
   FinSi
   Retourner y
FinFonction

// Début de l'algorithme
Début
   Ecrire("Entrez le nom, le prénom puis l'âge de l'étudiant 1 : ")
   Lire(etudiant1.nom, etudiant1.prénom, etudiant1.age)
   Ecrire("Entrez le nom, le prénom puis l'âge de l'étudiant 2 : ")
   Lire(etudiant2.nom, etudiant2.prénom, etudiant2.age)
   // Appel de la fonction Différence_age
   Ecrire("La différence d'âge entre ", etudiant1.nom, " et ",
   etudiant2.nom, " est :", Différence_age(Etudiant1, Etudiant2)
Fin
```