مكتب التكوين المهني وإنعاش الشغل
Office de la Formation Professionnelle et de la Promotion du Travail

Techniques De Programmation Structurée

Mme Amina NAINIA
Mme Amina NAINIA

Techniques de Développement informatique
Group « D »
Promotion 2010 | 2012

Sommaire

Les variables

Exercice 1 :

Debut

	A	B
A , B : entier	A	B
A ← 1	1	?
B ← A+3	1	4
A ← 3	3	4

Fin

Exercice 2 :

Debut

	A	B	C
A , B , C : entier	A	B	C
A ← 5	5	?	?
B ← 3	5	3	?
C ← A + B	5	3	8
A ← 2	2	3	8
C ← B – A	2	3	1

Fin

Exercice 3 :

Debut

	A	B
A , B , C : entier	A	B
A ← 5	5	?
B ← A + 4	5	9
A ← A + 1	6	9
B ← A – 4	6	2

Fin

Exercice 4 :

Debut

	A	B	C
A , B , C : entier	A	B	C
A ← 3	3	?	?
B ← 10	3	10	?
C ← A + B	3	10	13
B ← A + B	3	13	13
A ← C	13	13	13

Fin

Exercice 5 :

Debut

	A	B
A , B : entier	A	B
A ← 5	5	?
B ← 2	5	2
A ←B	2	2
B ←1	2	2

Fin

Exercice 6 :

Ecrire un algorithme qui permet d'échanger les valeurs de deux variables A & B.

```
Debut
        A , B , C :  entier
        Lire (A)
        Lire (B)
        C ← A
        A ← B
        B ← C
        Afficher (A)
        Afficher (B)
Fin
```

A	B	C
3	?	?
3	6	?
3	6	3
6	6	3
6	3	3
6		
3		

Exercice 7 :

On dispose de 3 variables A, B & C. Ecrire un algorithme qui permet de transformer à B la valeur de A, à C la valeur de B & à A la valeur de C.

```
Debut
        Variables : A , B , C , D : entier
        Lire (A)
        Lire (B)
        Lire (C)
        D ← B
        B ← A
        A ← C
        C ← D
        Afficher (A)
        Afficher (B)
        Afficher (C)
Fin
```

Exercice 8 :

Quel résultat produit l'algorithme suivant :

```
Debut
        Variables : double , val : entier
        val ← 231
        double ← val * 2
        Ecrire (val)
        Ecrire (double)
Fin
```

Ce algorithme afficher le double du valeur saisis par l'utilisateur

Exercice 9 :

Ecrire un algorithme qui demande à l'utilisateur un nombre, puis qui calcule et affiche le carré de ce nombre.

```
Debut
        Variables : nombre , carré : entier
        Lire (nombre)
        carré ← nombre * nombre
        Afficher ( "le carré de la valeur saisie est : ", carré)
Fin
```

Exercice 10 :

Ecrire un algorithme qui demande à l'utilisateur deux nombres, puis affiche leurs somme et leurs produit.

```
Debut
        Variables : nombre1 , nombre2 , somme , produit  : entier
        Lire (nombre1)
        Lire (nombre2)
        somme ← nombre1 + nombre2
        produit ← nombre1 * nombre2
        Afficher (somme)
        Afficher (produit)
Fin
```

Exercice 11 :

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

```
Debut
        Variables : prixHT , TVA , prixTTC : réel
                    nombrearticles : entier
        Lire (prixHT)
        Lire (TVA)
        Lire (nombrearticles)
        prixTTC ← (1 + TVA) * prixHT * nombrearticles
        Afficher (prixTTC)
Fin
```

Exercice 12 :

Ecrire un algorithme qui permet de calculer la surface d'un rectangle.

```
Debut
        Variables : longueur , largeur , surface : réel
        Afficher ( "Veuillez saisir la longueur : " )
        Lire (longueur)
        Afficher ( "Veuillez saisir la largeur :  " )
        Lire (largeur)
        Surface ← longueur * largeur
        Afficher (surface)
Fin
```

La structure alternative / Conditions

Exercice 1 :

Ecrire un algorithme qui lit 2 valeurs 'a' & 'b' et affiche la valeur la plus grande.

```
Debut
        Variables : a , b : entier
        Lire (a)
        Lire (a)
        Si (a > b) alors
                Afficher ("La valeur la plus grande est : ", a)
        SinonSi (a = b) alors
                Afficher ("Les deux valeurs sont égales.")
        Sinon
                Afficher ("La valeur la plus grande est : ", b)
        FinSi
Fin
```

Exercice 2 :

Ecrire un algorithme qui permet d'afficher la valeur absolue d'un entier saisi.

```
Debut
        Variables : a : entier
        Si (a < 0) alors
                Afficher ("La valeur absolue est : ", - a)
        Sinon
                Afficher ("La valeur absolue est : ", a)
        FinSi
Fin
```

Exercice 3 :

Ecrire un algorithme qui permet de résoudre une équation de $2^{ème}$ degré.

```
Debut
        Variables : a , b , c , D , x , x1 , x2 : entier
        Afficher ("Ce programme permet de résoudre une équation de 2ème degré.")
        Lire (a)
        Lire (b)
        Lire (c)
        Si (a <> 0) alors
                D ← b*b – 4*a*c
                Si (D > 0 ) alors
                        x1 ← (-b + sqrt(D)) / (2a)
                        x2 ← (-b – sqrt(D)) / (2a)
                        Afficher ("x1 = ", x1)
                        Afficher ("x2 = ", x2)
```

```
            SinonSi (D = 0) alors
                    x ← (– b) / (2a)
                    Afficher ("x = ", x)
            Sinon
                    Afficher ("Aucune solution. ")
             FinSi
        Sinon
             Si (b <> 0) alors
                     x ← (– c) / b
                     Afficher ("x = ", x)
             Sinon
                     Si (c <> 0) alors
                             Afficher ("Pas de solution. ")
                     Sinon
                             Afficher ("La solution est l'ensemble IR . ")
                     FinSi
             FinSi
         FinSi
Fin
```

Exercice 4 :

Ecrire un algorithme qui permet d'afficher le maximum de 3 nombres saisis.

```
Debut
        Variables : a , b , c : réel
        Lire (a)
        Lire (b)
        Lire (c)
        Si (a < b) alors
                Si (a < c) alors
                        Afficher ("La valeur la plus grande est : ", a)
                Sinon
                        Afficher ("La valeur la plus grande est : ", c)
                FinSi
        Sinon
                Si (b < c) alors
                        Afficher ("La valeur la plus grande est : ", b)
                Sinon
                        Afficher ("La valeur la plus grande est : ", c)
                FinSi
        FinSi
Fin
```

Exercice 5 :

Ecrire un algorithme qui permet de simuler le comportement d'une voiture devant un feu rouge.

```
Debut
        Variables : couleur : chaîne de caractère
        Lire (couleur)
        Si (couleur = "rouge") alors
                Afficher ("Stop !!!")
        SinonSi (couleur = "orange") alors
                Afficher ("Ralentir ! ")
        SinonSi (couleur = "vert")
                Afficher ("Démarrer … ")
        Sinon
                Afficher ("La couleur saisie n'est pas conforme au panneau de feu")
        FinSi
Fin
```

Exercice 6 :

Ecrire un algorithme qui lit 3 valeurs entre a , b & c et qui test si l'une de ces 3 valeurs est égale à la somme des 2 autres et l'affiche.

```
Debut
        Variables : a , b , c : entier
        Lire (a)
        Lire (b)
        Lire (c)
        Si (a = b+c) alors
                Afficher ("La somme des deux autres valeurs est : ", a)
        Sinon
                Si (c = a+b) alors
                        Afficher ("La somme des deux autres valeurs est : ", c)
                Sinon
                        Si (b = a+c) alors
                                Afficher ("La somme des deux autres valeurs est : ", b)
                        Sinon
                                Afficher ("Aucune valeur n'est la somme des deux autres. ")
                        FinSi
                 FinSi
        FinSi
Fin
```

Exercice 7 :

Ecrire un algorithme qui réalise une opération arithmétique de la façon suivante :

- ⇒ L'utilisateur produit le 1er argument du type numérique (entier,réel) qui présente le 1er nombre.
- ⇒ Il introduit le 2ème argument qui est du type caractère qui présente les signes d'opération (+,-,*,/)
- ⇒ Il introduit le 3ème argument du type numérique et qui présente le 2ème nombre.

```
Debut
        Variables : a , b , S : entier
                    signe : caractère
        Afficher ("Tapez le premier nombre : ")
        Lire (a)
        Afficher ("Tapez le signe d'opération : ")
        Lire (signe)
        Afficher ("Tapez le deuxième nombre : ")
        Lire (b)
        Si (signe = '+') alors
            S ← a + b
            Afficher (a , "+" , b , "=" , S)
        SinonSi (signe = '*') alors
            S ← a * b
            Afficher (a , "*" , b , "=" , S)
        SinonSi (signe = '–') alors
            Afficher (a , "–" , b , "=" , S)
        SinonSi (signe = '/') alors
            Si (b <> 0) alors
                S ← a / b
                Afficher (a , "/" , b , "=" , S)
            Sinon
                Afficher ("Division impossible." )
            FinSi

        Sinon
            Afficher ("Le signe ne correspond à aucune question")
        FinSi
Fin
```

Exercice 8 :

Ecrire un algorithme qui après avoir demandé un numéro de jour, de mois & d'année à l'utilisateur, revoie d'il s'agit d'une date valide ou non.

```
Debut
        Variables : J , M , A : entier
                    B : booléenne
        Afficher ("Veuillez saisir le jour, le mois & l'année : ")
        Lire (J)
        Lire (M)
        Lire (A)
        Si ( (M>12) ou (M<1) ou (J<1) ou (J>31) ou (A<=0) ) alors
            Afficher ("Date non valide")
        Sinon
            Si ( (M=1) ou (M=3) ou (M=5) ou (M=7) ou (M=8) ou (M=10) ou (M=12) ) alors
                Afficher ("Date valide! ")
            Sinon
                Si ( (M=4) ou (M=6) ou (M=9) ou (M=11) ) alors
                    Si (J<=30) alors
                        Afficher ("Date valide !" )
                    Sinon
```

```
                        Afficher ("Date non valide")
                    FinSi
                Sinon
                    Si (M=2) alors
                        Si ( (A%400 = 0) ou ((A%4 = 0) et (A%100 <> 0)) ) alors
                            Si (J <= 29) alors
                                Afficher ("Date valide !" )
                            Sinon
                                Afficher ("Date non valide")
                            FinSi
                        Sinon
                            Si (J <= 28) alors
                                Afficher ("Date valide !" )
                            Sinon
                                Afficher ("Date non valide")
                            FinSi
                        FinSi
                    FinSi
                FinSi
            FinSi
        FinSi
Fin
```

Exercice 9 :

Ecrire un algorithme qui à partir d'un nombre compris entre 1 et 7 affiche le jour correspondant.

```
Debut
    Variables : n : entier
    Afficher ("Saisir un nombre entre 1 et 7 : ")
    Lire (n)
    Si ( (n<1) et (n>7)) alors
        Afficher ("Erreur !" )
    SinonSi (n = 1) alors
        Afficher (n, " = Lundi")
    SinonSi (n = 2) alors
        Afficher (n, " = Mardi")
    SinonSi (n = 3) alors
        Afficher (n, " = Mercredi")
    SinonSi (n = 4) alors
        Afficher (n, " = Jeudi")
    SinonSi (n = 5) alors
        Afficher (n, " = Vendredi")
    SinonSi (n = 6) alors
        Afficher (n, " = Samedi")
    SinonSi (n = 7) alors
        Afficher (n, " = Dimanche")
Fin
```

La Structure Alternative « Cas Parmi »

Exercice 1 :

Ecrire un algorithme qui permet, à partir d'une moyenne annuelle entière, de déterminer l'appréciation sur les résultats des étudiants.
Si la moyenne est 0 , 1 , 2 , 3 , 4 , 5 : l'avis est nul.
Si la moyenne est 6 ou 7 : l'avis est très insuffisant.
Si la moyenne est 8 ou 9 : l'avis est insuffisant.
Si la moyenne est 10 ou 11 : l'avis est moyen.
Si la moyenne est 12 ou 13 : l'avis est assez bien.
Si la moyenne est 14 ou 15 ou 16 : l'avis est bien.
Si la moyenne est 17 ou 18 ou 19 : l'avis est très bien.

```
Debut
        Variables : moyenne : entier
        Lire (moyenne)
        Cas moyenne parmi
        [0,5] : Afficher ("nul")
        [6,7] : Afficher ("très insufficant")
        [8,9] : Afficher ("insuffisant")
        [10,11] : Afficher ("moyen")
        [12,13] : Afficher ("assez bien")
        [14,16] : Afficher ("bien")
        [17,19] : Afficher ("très bien")
        Autre  : Afficher ("Erreur !")
        FinCas
Fin
```

Exercice 2 :

Un patron décide de calculer le montant de sa participation au repas de 16 employés de la façon suivante :

- ⇒ Si l'employé est célibataire, la participation est de 20%
- ⇒ Si l'employé est marié, la participation est de 25%
- ⇒ Si l'employé a des enfants, la participation est de 10% par enfant.
- ⇒ La participation est plafonné de 50%
- ⇒ Si le salaire est <6000 la participation est majorée de 10%

Ecrire un algorithme qui lit les informations des employés et calcule et affecte la participation du patron.

```
Debut
        Variables : sitfamilial : chaîne de caractères
                    salaire , prix , P : réel
                    Nbrenfant : entier
        Afficher ("Veuillez saisir le prix de repas : ")
        Lire (prix)
        Afficher ("Veuillez saisir les informations de l'employé :" )
        Lire (sitfamilial)
        Lire (Salaire)

        Cas sitfamilial parmi
            "celibataire" : P ← prix * 20%
            "marie"  :  P ← prix * 25%
                        Afficher ("Veuillez saisir le nombre d'enfants, 0 si pas d'enfants")
                        Lire (nbrenfant)
                        P ←  P * nbrenfant * prix * 10%
        FinCas
        Si (P > 50%*prix) alors
              P ←  prix * 50%
        FinSi
        Si (salaire < 6000) alors
              P ←  P + prix*10%
        FinSi
        Afficher ("La participation du patron est : ", P)
Fin
```

La Structure Alternative « TantQue »

Exercice 1 :

Ecrire un algorithme qui permet de calculer la somme de n premiers entiers avec n saisis par l'utilisateur.

```
Debut
        Variables :  i , n , S : entier
        Lire (n)
        Tantque (n < 0) faire
              Lire (n)
        FinTantque
        i ← 1
        S ← 0
        Tantque (i <= n) faire
               S ← S + i
               i ← i + 1
        FinTantque
        Afficher (S)
Fin
```

Exercice 2 :

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

```
Debut
        Variables :   n: entier
        Lire (n)
        Tantque ((n<0) ou (n>3)) faire
              Lire (n)
        FinTantque
Fin
```

Exercice 3 :

Ecrire un algorithme qui demande à l'utilisateur un nombre compris entre 10 et 20, jusqu'à ce que la réponse convienne. En cas de réponse supérieur à 20, on fera apparaître un message « plus petie » et inversement « plus grand » si le nombre est inférieur à 10.

```
Debut
        Variables :   n: entier
        Lire (n)
        Tantque ((n<10) ou (n>20)) faire
              Si (n<10) alors
                    Afficher ("Plus grand")
              Sinon
                    Afficher ("Plus petit")
              Finsi
              Lire (n)
        FinTantque
        Afficher (n)
Fin
```

Exercice 4 :

Ecrire un algorithme qui demande à l'utilisateur un nombre de départ, et qui ensuite affiche les dix nombres suivants.

```
Debut
        Variables :   i , n: entier
        Ecrire ("Saisir un nombre : ")
        Lire (n)
        Tantque (i <= 10) faire
                n ← n + 1
                i ← i + 1
                Afficher (n)
        FinTantque
Fin
```

Exercice 5 :

Ecrire un algorithme qui demande un nombre de départ et qui ensuite ecrit la table de multiplication de ce nombre, présentée comme suite Table de 7 : 7 x 1 = 7
7 x 2 = 14
……..

```
Debut
        Variables :   i , m , n: entier
        Ecrire ("Veuillez saisir un nombre de départ : ")
        Lire (n)
        i ← 1
        Tantque (i <= 10) faire
                M ← n * i
                Afficher (n , "*" , i , "=" , m)
                i ← i + 1
        FinTantque
Fin
```

Exercice 6 :

Ecrire un algorithme qui lit un nombre « x », puis un entier « n » puis calcule et affiche la puissance n-ième de x^n.

```
Debut
        Variables :   i , m , n: entier
        Ecrire ("Veuillez saisir un nombre de départ : ")
        Lire (n)
        i ← 1
        Tantque (i <= 10) faire
                m ← n * i
                Afficher (n , "*" , i , "=" , m)
                i ← i + 1
        FinTantque
Fin
```

Exercice 7 :

Ecrire un algorithme qui lit un nombre x, puis un entier n, puis calcule et affiche la puissance n-ième de x^n .

```
Debut
        Variables :   x , n , P , m  : entier
        Lire (x)
        Lire (n)
        i ← 1
        Si (n > 0) alors
                m ← n
        Sinon
                m ← – n
        FinSi
        i ← i + 1
        P ← 1
        Tantque (i <= m) faire
                P ← x + P
                i ← i + 1
        FinTantque
        Si (n > 0) alors
                Afficher (x , " puissance " , n , "=" , P)
        Sinon
                Si (x <> 0) alors
                        Afficher (n , " puissance " , n , "=" , 1/P)
                Sinon
                        Afficher (" Erreur !!!" )
                FinSi
        FinSi
Fin
```

LA STRUCTURE ALTERNATIVE « RÉPÉTER … JUSQU'À »

Exercice 1 :

Ecrire un algorithme qui demande à l'utilisateur de saisir 20 nombres et qui lui dise ensuite quel était le grand parmi ces 20 nombres.

```
Debut
        Variables :   i , n , max  : entier
        Afficher ("Saisir le nombre numéro 1 : ")
        Lire (n)
        max ← n
        i  ← 2
        Répéter
                Afficher ("Saisir le nombre n° " , i)
                Lire (n)
                Si (n > max) alors
                        max ←  n
                FinSi
                i  ← i + 1
        Jusqu'à (i > 20)
        Afficher ("La valeur la plus grande est " , max)
Fin
```

Exercice 2 :

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

```
Debut
        Variables :   i , n , max  : entier
        Afficher ("Saisir le nombre numéro 1 : ")
        Lire (n)
        max  ← n
        i  ←  2
        Répéter
                Afficher ("Saisir le nombre n° " , i)
                Lire (n)
                Si (n > max) alors
                        max ← n
                FinSi
                i  ←  i + 1
        Jusqu'à (n = 0)
        Afficher ("La valeur la plus grande est : " , max)
Fin
```

Exercice 3 :

Ecrire un algorithme qui calcule la moyenne d'une série de nombres entiers positifs ou nuls, lus au clavier et l'affiche à l'écran. Le programme s'arrête dès qu'on introduit un nombre négatif. On pourra supposer qu'il y a au moins un nombre positif dans la scène.

```
Debut
        Variables :   i , n , S : entire
                      M : réel
        S ← 0
        i ←  0
        Répéter
              i  ←  i + 1
              Afficher ("Saisir le nombre n° " , i)
              Lire (n)
              S ←  S + n
        Jusqu'à (n < 0)
        M ←  S / i
        Afficher (M)
Fin
```

Exercice 4 :

Ecrire un algorithme qui vérifie si un entier est premier ou non.

METHODE 1:

```
Debut
        Variables :   i , n  : entire
        Afficher ("Saisir un nombre : ")
        Répéter
              Lire (n)
        Jusqu'à (n > 0)
        i  ← 1
        Répéter
              i  ←  i + 1
        Jusqu'à ( (n % i = 0) ou (i > √n ) )
        Si (n % i = 0) alors
              Afficher ("Le nombre " , n , " n'est pas premier.")
        Sinon
              Afficher ("Le nombre " , n , "est premier." )
        FinSi
Fin
```

METHODE 2:

```
Debut
        Variables :   n , i , SDN  : entire
        Afficher ("Saisir un entier : ")
        Répéter
              Lire (n)
        Jusqu'à (n > 1)
        SDN ← 0
```

```
        i ← 1

        Tantque (i <= n) faire
                Si (n mod i = 0) alors
                        SDN ←  SDN + 1
                 FinSi
                 i ← i + 1
        FinTantque
        Si (SDN = n + 1) alors
                Afficher ("Cet entier est premier.")
        Sinon
                Afficher ("Cet entier n'est pas premier." )
        FinSi
Fin
```

METHODE 3:

```
Debut
        Variables :   n , i  : entire
                      Premier  : booleen
        Afficher ("Saisir un entier : ")
        Répéter
                Lire (n)
        Jusqu'à (n > 0)
        Premier ← vrai
        i ← 2
        Tantque ( (Premier = vrai) et (i <= √n ) faire
                Si (n % i = 0) alors
                     premier ← faux
                Sinon
                      i  ← i + 1
                FinSi
        FinTantque
        Si (premier = vrai) alors
                Afficher (n , " est premier. ")
        Sinon
                Afficher (n , " n'est pas premier. ")
        FinSi
Fin
```

Exercice Nbr Parfait :

```
Debut
        Variables :   n , i  : entire
                      Premier  : booleen
        Afficher ("Saisir un entier : ")
        Répéter
                Lire (n)
        Jusqu'à (n > 0)
        Premier ← vrai
        i ← 2
        Tantque ( (Premier = vrai) et (i <= √n ) faire
                Si (n % i = 0) alors
                     premier ← faux
                Sinon
                      i  ← i + 1
                FinSi
        FinTantque
        Si (premier = vrai) alors
                Afficher (n , " est premier. ")
        Sinon
                Afficher (n , " n'est pas premier. ")
        FinSi
Fin
```

Le Factoriel

Exercice 1 :

Calculer la série : $e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + ... + \frac{1}{n!}$

```
Debut
        Variables :   n , i , F  : entire
                      S : reel
        Lire (n)
        Tantque (n < 1) faire
              Lire (n)
        FinTantque
        S ← 1
        F ← 1
        Pour   i ←  1 à  n  faire
              F ← F * i
              S ← S + 1 / F
        FinPour
        Afficher (S)
Fin
```

Exercice 2 :

Calculer la série : $e = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + ... + \frac{x^n}{n!}$

```
Debut
        Variables :   n , i , x , F , P , k  : entire
                      S : reel
        Lire (n)
        Tantque (n % 2 = 0) faire
              Lire (n)
        FinTantque
        Lire (x)
        k ←  1
        F ←  1
        P ←  1
        S ←  0
        Pour   i ←1 à  n  faire
              F ← F * i
              P  ← P * x
              Si (i % 2 <> 0) alors
                    S ← S + k * (P/F)
                    k ←  – k
              FinSi
        FinPour
        Afficher (S)
Fin
```

Les Tableaux
Activité s'apprentissage de base n° b.13

Exercice 1 :

Ecrire un algorithme qui affiche la valeur maximale et sa position dans un tableau de 20 entiers saisis.

```
Debut
        Variables :   i , max , pos , Tab[20]  : entire
        Lire (tab[i])
        max ← Tab[0]
        pos ← 0
        Pour  i ← 1 à  19  faire
                Lire ( Tab[i] )
                Si (Tab[i] > max) alors
                    max ← Tab[i]
                    pos ← i
                FinSi
        FinPour
        Afficher ("La valeur maximale est " , max)
        Afficher ("Sa position est " , pos)
Fin
```

Exercice 2 :

Soit un tableau d'entier de taille 10. Proposer un algorithme qui permet de calculer la somme des éléments de ce tableau.

```
Debut
        Variables :   i , S , Tab[10]  : entire
        S ← 0
        Pour  i ← 1 à  9  faire
                Lire ( Tab[i] )
                S ← S + Tab[i]
        FinPour
        Afficher (S)
Fin
```

Exercice 3 :

Soit un tableau d'entier de taille 10. Proposer un algorithme qui permet de lire le contenu du tableau et affiche pour chaque case sa factorielle.

```
Debut
        Variables :   i , F ,  j  , Tab[10]  : entire
        Pour   i ← 1 à  9  faire
                F ←  1
                Répéter
                      Lire ( Tab[i] )
                Jusqu'à (Tab[i] > 0)



                Pour   j ← 1 à  Tab[i]  faire
                          F ← F * j
                FinPour
        Afficher ("Le factoriel de " , Tab[i] , " est  " , F)
Fin
```

Exercice 4 :

Ecrire un algorithme qui permet de stocker 10 valeurs dans un tableau d'entiers. Il doit afficher le nombre de valeurs positifs, nuls ou négatifs.

```
Debut
         Variables :  i , Pos , Neg , Nul , Tab[i] : entier
          Pos ←  0
          Neg ← 0
          Nul  ←  0
          Pour   i ← 0  à  9 faire
                    Lire (Tab[i])
                    Si ( Tab[i] > 0) alors
                            Pos ← Pos + 1
                    Sinon
                            Si (Tab[i] < 0) alors
                                    Neg ← Neg + 1
                            Sinon
                                    Nul ← Nul + 1
                            FinSi
                    FinSi
          FinPour
          Afficher ("Le nombre de valeurs positifs est : " , Pos)
          Afficher ("Le nombre de valeurs négatifs est : ", Neg)
          Afficher ("Le nombre de valeurs nuls est : " , Nul)
Fin
```

Exercice 5 :

Soit un tableau d'entiers de taille 10. Proposer un algorithme qui permet de remplir les cases de tableau avec des valeurs strictement positives, et afficher par la suite un histogramme où le nombre des étoiles de la ligne i est T[i].

```
Debut
        Variables :  i , j , Tab[10] : entier
        Pour  i←1 à 9 faire
                Répéter
                        Lire (Tab[i])
                Jusqu'à (Tab[i] > 0)
                Afficher ("Pour la ligne " , i + 1 , ": ")
                Pour  j←1  à  Tab[i] faire
                        Afficher ("*")
                FinPour
                // Traduire un retour à la ligne
        FinPour
Fin
```

Exercice 6 :

Proposer un algorithme qui permet de stocker dix valeurs dans un tableau d'entiers et de permuter la dernière case avec la première, l'avant dernière avec la deuxième, et ainsi de suite. Affiche le résultat final.

```
Debut
        Variables :  i , j , Tab[i] , x   : entier
        Pour  i←0  à  9 faire
                Afficher ("Veuillez saisir la valeur : " , i + 1)
                Lire (Tab[i])
        FinPour
        i ← 0
        j ← 9
        Tantque (i < j) faire
                x ← Tab[i]
                Tab[i] ← Tab[j]
                Tab[j] ← x
                i ← i + 1
                j ← j – 1
        FinTantque
Fin
```

Exercice 7: recherche séquentielle

Ecrire un algorithme qui remplit un tableau des éléments et qui cherche si un élément existe. Si oui, on affiche sa position. Sinon, on affiche un message.

```
Debut
    Variables :  i , val , Tab[i] : entier
                 R : booleen
    Pour i←0  à  9  faire
        Ecrire ("Veuillez saisir le nombre n° " , i + 1)
        Lire (Tab[i])
    FinPour
    Ecrire ("Veuillez saisir la valeur : ")
    Lire (val)
    i ← 0
    R ← faux
    Répéter
        Si (val = Tab[i]) alors
            R ← vrai
        Sinon
            i ← i + 1
        FinSi
    Jusqu'à ( (R = vrai) ou (i > 9) )
    Si (R = vrai ) alors
        Ecrire ("Cette valeur existe, sa position est : " , i + 1)
    Sinon
        Ecrire ("Cette valeur n'existe pas ! ")
    FinSi
Fin
```

Exercice 8 :

Ecrire un algorithme qui permet de déterminer si un mot saisi est un palindrome.

```
Debut
    Variables : i , j , n , Tab[30] : entier
                R : booleen
    Ecrire ("Veuillez saisir un mot, pour terminer appuyer sur point .")
    i ← 0
    Répéter
        Lire (Tab[i])
        i ← i + 1
    Jusqu'à ( (Tab[i] = ".") ou (i > 30) )
    n ← i
    i ←0
    j ← n – 1
    R ← vrai
```

```
        Tantque ( (R = vrai) et (i < j) ) faire
                Si (Tab[i] = Tab[j]) alors
                        i ← i + 1
                        j ← j – 1
                Sinon
                        R ← faux
                FinSi
        FinTantque
        Si (R = vrai) alors
                Afficher ("Le mot est palindrome")
        Sinon
                Afficher ("Le mot n'est pas palindrome")
        FinSi
Fin
```

Exercice 9 :

Ecrire un algorithme qui permet de construire le mot miroir d'un mot saisi.

```
Debut
        Variables :  i , j , n , Tab[16] , m : entier
        Répéter
                Lire (n)
        Jusqu'à (n >= 0)
        Pour i←0  à  15 faire
                Tab[i] ← 0
        FinPour
        Répéter
                Tab[i] ← n % 2
                n ← n / 2
                i ← i + 1
        Jusqu'à (n = 0)
        j ← i – 1
        i ← 0
        Tantque (i < j) faire
                m ← Tab[i]
                Tab[i] ← tab[j]
                Tab[j] ← m
                i ← i + 1
                j ← j – 1
        FinTantque
        Pour i←0  à  15 faire
                Afficher (Tab[i])
        FinPour
Fin
```

Les Matrices
Activité d'apprentissage de base n° b.13

Exercice 1 :

Ecrire l'algorithme qui calcule et affiche la somme de deux matrices.

```
Debut
        Variables :  i , j , mat1(10,10) , mat2(10,10) , mat3(10,10) : entier
        Pout i←0  à  9 faire
                Pour j←0 à  9 faire
                        Lire (mat1(i,j))
                FinPour
        FinPour
        Pour i←0  à  9 faire
                Pour j←0  à  9  faire
                        Lire (mat2(i,j))
                FinPour
        FinPour
        Afficher ("La somme des deux matrices est :  ")
        Pour i←0  à  9  faire
                Pour j←0  à  9  faire
                        mat3(i,j) ← mat1(i,j) + mat2(i,j)
                        Afficher (mat3(i,j))
                FinPour
        FinPour
Fin
```

Exercice 2:

Soit une matrice de 5 lignes et 5 colonnes. Ecrire un algorithme qui permet de remplir les cases de la diagonale par 1 et les autres cases par 0.

```
Debut
        Variables :  i , j , mat(5,5) : entier
        Pour i←0  à  4 faire
              Pour j←0  à  4 faire
                    Si (i = j) alors
                          mat(i,j) ← 1
                    Sinon
                          mat(i,j) ← 0
                    FinSi
              FinPour
        FinPour
Fin
```

Exercice 3 :

Soit une matrice de 5 lignes et 5 colonnes. Ecrire un algorithme qui permet de remplir les cases de la diagonale par 1, les cases qui sont au-dessus de la diagonale par 0 et les cases qui sont au dessous de la diagonale par -1.

```
Debut
        Variables :  i , j , mat(5,5)  : entire
        Pour i←0  à  4 faire
              Pour j←0  à  4 faire
                    Si (i = j) alors
                          mat(i,j) ← 1
                    Sinon
                          Si (i < j) alors
                                mat(i,j) ← 0
                          Sinon
                                mat(i,j) ← – 1
                          FinSi
                    FinSi
              FinPour
        FinPour
        Pour i←0  à  4  faire
              Pour j←0  à  4 faire
                    Afficher (mat(i,j))
              FinPour
        FinPour
Fin
```

Exercice 4 :

Soit une matrice de 5 lignes et 5 colonnes. Ecrire l'algorithme qui permet de remplir les cases au dessus de la ligne du milieu par 1 et celles au dessous par -1.

```
Debut
        Variables :  i , j , mat(5,5) : entier
        Pour i←0  à  4 faire
              Pour j←0  à  4 faire
                    Si (i > 2) alors
                          mat(i,j) ← – 1
                    Sinon
                          Si (i < 2) alors
                                mat(i,j) ← 1
                          Sinon
                                mat(i,j) ← 0
                          FinSi
                    FinSi
              FinPour
        FinPour
        //2 boucles pour l' affichage de la matrice.
Fin
```

Exercice 5 :

Proposer un algorithme qui permet de vérifier si une matrice est unitaire.

```
Debut
        Variables :   mat(5,5) , i , j : entier
                      B : booleen
        Pour i←0  à  4 faire
              Pour j←0  à  4 faire
                    Lire (mat(5,5))
              FinPour
        FinPour
        i ← 0
        B ← vrai
        Répéter
              j ← 0
              Répéter
                    Si ( (i = j) et (mat(i,j) <> 1) ) alors
                          B ← faux
                    FinSi
                    Si ( (i <> j) et (mat(i,j) <> 0) ) alors
                          B ← faux
                    FinSi
                    j ← j + 1
              Jusqu'à ( (B = faux) ou (j = 5) )
              i ← i + 1
        Jusqu'à ( (B = faux) ou (i = 5) )
        Si (B = faux) alors
              Afficher ("La matrice n'est pas unitaire. ")
        Sinon
              Afficher ("La matrice est unitaire." )
        FinSi
Fin
```

Exercice 6 :

Proposer un algorithme qui permet de vérifier si une matrice est symétrique par rapport à la diagonale.

```
Debut
        Variables : mat(5,5) , i , j : entier
                    B : booleen
        Pour i←0  à  4  faire
              Pour j←0  à  4 faire
                    Lire (mat(i,j))
              FinPour
        FinPour
        B ← vrai
        i ← 0
        Répéter
              Répéter
```

```
                    Si (i > j) alors
                        Si ( (mat(i,j) <> mat(j,i) ) alors
                            B ← faux
                        FinSi
                    FinSi
                    j ← j + 1
                Jusqu'à ( (B = faux) ou (j = 5) )
                i ← i + 1
        Jusqu'à ( (B = faux) ou (i = 5) )
        Si (B = faux) alors
            Afficher ("La matrice n'est pas symétrique." )
        Sinon
            Afficher ("La matrice est symétrique." )
        FinSi
Fin
```

Exercice 7 :

Ecrire l'algorithme qui lit une matrice et affiche sa transposée.

```
Debut
        Variables :  mat(5,5) , i , j , T(5,5) : entier
        Pour i←0  à  4  faire
            Pour j←0  à  4  faire
                Lire (mat(i,j))
            FinPour
        FinPour
        Pour i←0  à  4  faire
            Pour j←0  à  4  faire
                T(i,j) ← mat(j,i)
            FinPour
        FinPour
        Pour i←0 à 4  faire
            Pour j←0 à 4  faire
                Afficher (T(i,j))
            FinPour
        FinPour
Fin
```

Exercice 8 :

Ecrire un algorithme qui permet de réaliser la multiplication de deux matrices.

```
Debut
        Variables :  i , k ,  j , mat(4,4) , mat2(4,4) , mat3(4,4)  : entire
        Pour i←0 à 3  faire
              Pour j←0  à  3 faire
                    Lire (mat(i,j))
              FinPour
        FinPour
        Pour i←0 à  3  faire
              Pour j←0  à  3 faire
                    Lire (mat2(i,j))
              FinPour
        FinPour
        Pour i←0 à  3  faire
              Pour j←0  à  3 faire
                    mat(i,j) ← 0
                    Pour k←0  à  3  faire
                          mat3(i,j) ← mat3(i,j) + mat(i,j)*mat2(i,j)
                    FinPour
              FinPour
        FinPour
        Pour i←0  à  3  faire
              Pour j←0  à  3 faire
                     Afficher (mat3(i,j))
              FinPour
        FinPour
Fin
```

Les Tris
Activité d'apprentissage code D004

Exercice 1 : Permutation

```
Debut
        Variables : i , j , k , p , Tab[10] : entier
        Pour i←0 à 9  faire
                Lire (Tab[i])
        FinPour
        Pour i←0  à  8 faire
                Si (Tab[i + 1] < Tab[i]) alors
                        p ← Tab[i + 1]
                        j ← 0
                        Tantque ( (j < i) et (Tab[ j] < Tab[i +1]) ) faire
                                j ← j + 1
                        FinTantque
                        Pour k←i + 1  à  j +1  faire (pas = -1)
                                Tab[k] ← Tab[k – 1]
                        FinPour
                        Tab[j] ← p
                FinSi
        FinPour
        Pour i←0  à  9  faire
                Afficher (Tab[i])
        FinPour
Fin
```

Exercice 2 : Comptage

```
Debut
        Variables : i , j , Res[10] , Nb[10] , Tab[10] : entier
        // Lecture du tableau.
        Pour i←0  à  9 faire
                Res[i]←0
                Nb[i] ←0
                Pour j←0 à 8 faire
                        Si (Tab[j] < Tab[i]) alors
                                Nb[i] ← Nb[i] + 1
                        FinSi
                FinPour
                Pour i←0 à  9 faire
                        J ←Nb[i]
                        Tantque (Res [i] <> 0) faire
                                j ← j +1
                        FinTantque
                        Res[j] ← Tab[i]
                FinPour
Fin
```

Les Tris
Activité d'apprentissage code D003

Exercice 1 : Sélection

```
Debut
        Variables :  i , j , Min , Pos , Tab[10] : entier
        Pour i←0 à  9  faire
              Lire (Tab[i])
        FinPour
        Pour i←0  à  8  faire
              Min ← Tab[i]
              Pos ← i
              Pour j ←i+1  à  9  faire
                    Si (Tab[j] < Min) alors
                          Min ← Tab[j]
                          Pos ← j
                     FinSi
              FinPour
              Si (Pos <> i) alors
                    ¨Pour j←Pos  à  i+1  (pas= -1) faire
                          Tab[j] ← Tab[j – 1]
                     FinPour
                     Tab[i] ← Min
              FinSi
         FinPour
         Pour i ←0  à  9  faire
               Afficher (Tab[i])
         FinPour
Fin
```

La Recherche dichotomique

```
Debut
        Variables : i , Tab[10] , debut , fin , val , m : entier
                    B : booleen
        Pour i←0  à  9  faire
              Lire (Tab[i])
        FinPour
        Afficher ("Donner la valeur à rechercher : ")
        Lire (val)
        B ← faux
        debut ← 0
        fin ← 9
        Répéter
              m ← (début + fin)/2
              Si (val = Tab[m]) alors
                    B ← vrai
              Sinon
                    Si (val > Tab[m]) alors
                          d ← m + 1
                    Sinon
                           f ← m – 1
                    FinSi
              FinSi
        Jusqu'à  ( (B = vrai) ou (d > f) )
        Si (B = vrai) alors
              Afficher ("La valeur existe dans le tableau." )
        Sinon
              Afficher ("La valeur n'existe pas dans le tableau." )
        FinSi
Fin
```

Les Procédures et Les Fonctions

Introduction :

Cas 1 :

Ecrire un algorithme qui produit l'affichage suivant :

```
* * * * * * * *
Bonjours tout le monde
* * * * * * * *
On est le 5 / 1 / 2011
* * * * * * * *
Bonne journée
```

```
Debut
        Variables :  i : entier
        Pour i←0  à  7  faire
              Afficher ("*")
        FinPour
        Afficher ("Bonjours tout le monde")
        Pour i←0  à  7  faire
              Afficher ("*")
        FinPour
        Afficher ("On est le 5 / 1 / 2011)
        Pour i←0  à  7  faire
              Afficher ("*")
        FinPour
        Afficher ("Bonne journée")
Fin
```

Remarque :

Dans cette algorithme on remarque que la boucle «A» se répète plusieurs fois.

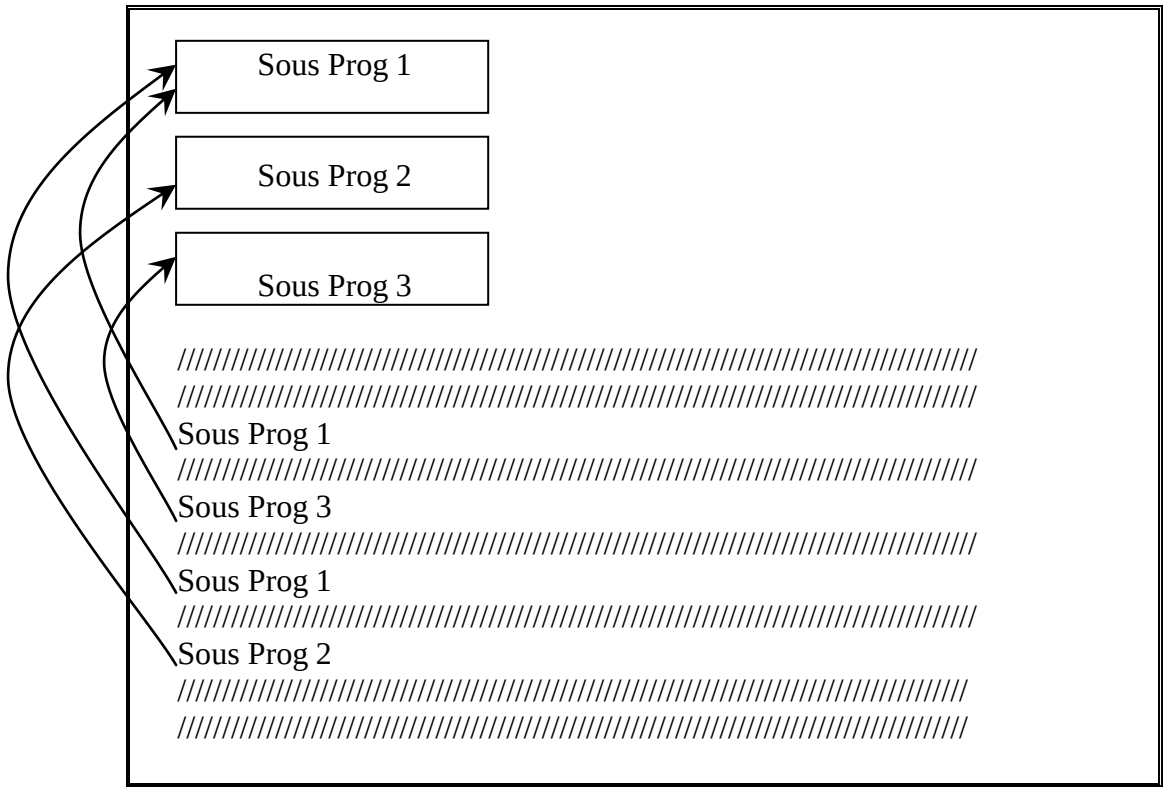
Cas 2 :

On a un algorithme compliqué qui traite plusieurs sujets.
La solution est de décomposer l'algorithme en parties qu'on appel sous programme.

***Sous programme** : un regroupement de données et d'instructions auxquels on donne un nom et qu'on peut utiliser (Appeler) dans un autre programme (Le programme appelant ou le programme principal).*

Les avantages des sous programmes :

- ⇒ Eviter la répétition.
- ⇒ Structurer le programme.
- ⇒ Réutiliser plusieurs fois le même code.



1 – Les Procédures :

Une procédure est un sous programme dont le nom ne retourne pas de valeur.

Syntaxe :

Définition de la procédure :

```
Procédure        NomProcédure ( )
  Début
    |
    |            Déclaration de variables
    |             Instructions
    |
  Fin
```

Appel de la procédure : **NomProcédure ( )**

Exemple : Procédure qui affiche un message.

```
Procédure        AfficherMessage ( )
  Début
    |            Afficher ("Bonjour tout le monde")
    |
  Fin
Début
    |            Ecrire ("Test 1")
    |            AfficherMessage ( )
    |            Ecrire ("Test 2")
    |
Fin
```

Exercice 1 :

Reprendre l'exercice précédent (Cas 1) en écrivant une procédure qui affiche les * et faire son appel dans le programme principal.

```
Procédure       AfficherEtoile ( )
  Debut
    |
    |           Variables : i : entier
    |           Pour i←0  à  7  faire
    |                 Afficher ("*")
    |           FinPour
  Fin
Debut
    |           AfficherEtoile ( )
    |           Afficher ("Bonjours tout le monde")
    |           AfficherEtoile ( )
    |           Afficher ("On est le 5 / 1 / 2011")
    |           AfficherEtoile ( )
    |           Afficher ("Bonne journée")
Fin
```

Exercice 2 :

Ecrire une procédure qui affiche les 10 premier nombres et faire son appel dans le programme principal.

```
Procédure       AfficherNombre ( )
  Debut
    |
    |           Variables : i : entier
    |           Pour i←1  à  10  faire
    |                 Afficher (i)
    |           FinPour
  Fin
Debut
    |           Afficher ("Les 10 premier nombres sont :")
    |           AfficherNombre ( )
Fin
```

Exercice 3 :

Procédure qui calcule et affiche la somme de 2 entiers saisis par l'utilisateur.

```
Procédure       AfficherSomme ( )
  Debut
    |
    |           Variables : a , b , s : entier
    |           Lire (a)
    |           Lire (b)
    |           S← a + b
    |           Afficher (S)
  Fin
Debut
    |           Afficher ("La Somme des deux nombres est :")
    |           AfficherSomme ( )
Fin
```

Exercice 4 :

Procédure qui calcule et affiche le factoriel d'un nombre saisi par l'utilisateur avec son appel dans le programme principal.

```
Procédure       Factoriel ( )
  Debut
    |           Variables : i , n , F : entier
    |           Lire (n)
    |           TantQue (n<=0) faire
    |                 Lire (n)
    |           FinTantQue
    |           F ← 1
    |           Pour i←1  à  n  faire
    |                 F ← F * i
    |           FinPour
    |           Afficher ("Le factoriel est :  ")
  Fin
Debut
  |           Factoriel ( )
Fin
```

2 – Les Fonctions :

Une fonction est un sous programme dont le nom retourne une valeur de même type (entier, réel, chaîne de caractère…) que la fonction.

1) Définition de la fonction :

```
Fonction        NomFonction ( ) : TypedeRetour
 Début
   |
   |            Déclaration de variables
   |             Instructions
   |             Retourne Expression
  Fin
```

Remarque :

⇒ Expression à la même type que la fonction.

⇒ Expression peut être une valeur, une variable ou bien une expression arithmétique ou logique.

2) Appel de fonction :

```
Var1 : TypedeRetour
Var1 ← NomFonction ( )
```

Exemple :

```
Fonction        Somme ( ) : entier
  Début
    |
    |           x , y , S : entier
    |           Lire (x)
    |           Lire (y)
    |           S ← x + y
    |           Retourne  S
  Fin
Début
  |
  |           S1 : entier
  |           S1 ← Somme ( )
  |           Afficher(S1)
Fin
```

Exercice 1 :

Une fonction qui calcule et affiche le produit de 2 entiers.

```
Fonction        Produit ( ) : entier
  | Debut
  |   |
  |   |       Variables : a , b , P : entier
  |   |       Afficher ("Veuillez saisir le 1er nombre : ")
  |   |       Lire (a)
  |   |       Afficher ("Veuillez saisir le 2ème nombre : ")
  |   |       Lire (b)
  |   |       P ← a * b
  |   |       Retourne  P
  | Fin
Debut
  |
  |           k : entier
  |           k ← Produit ( )
  |           Afficher ( k)
Fin
```

Exercice 2 :

Une fonction qui calcule la factoriel d'un entier saisis par l'utilisateur. Le résultat s'affiche dans le programme principal.

```
Fonction        Factoriel ( ) : entier
  Debut
    |
    |           Variables : i , n , F  : entier
    |           Répéter
    |                 Lire (n)
    |           Jusqu'à (n > 0)
    |           F ← 1
    |           Pour i←1  à  n  faire
    |                 F ← F * i
    |           FinPour
    |           Retourne  F
  Fin
Debut
  |           a : entier
  |           a ← Factoriel ( )
  |           Afficher ( a)
Fin
```

3 - La portée de variables :

- Variables locales :

Soit la procédure et le programme principal suivant :

```
Procédure       AfficherMessage ( )
  Début
    |
    |           i : entier
    |           i ← 1
    |           Afficher (i)
    |           i ← i + 1
  Fin
Début
  |           Test1 ( )
  |           Ecrire (i)
Fin
```

Le compilateur affiche une **erreur** car « i » non déclaré dans le programme principal. Il est déclaré dans la procédure.
On dit que « i » une **variable locale** à la procédure.

- **Variables globales :**

```
x : entier
Procédure        Test1 ( )
   Début
                 x ← x + 1
                 Afficher (x)
   Fin
Début
                 x ← 3
                 Afficher (x)
                 Test1 ( )
                 Afficher (x)
Fin
```

Ce code produit l'affichage suivant :

```
3
4
4
```

Car « x » est une variable globale donc il est visible dans la procédure test et dans le programme principal.

4 - Le passage de paramètre :

Si on a besoin d'une variable dans le programme principal et dans la procédure on utilise le passage de paramètre.
Les paramètres constituent le moyen qui nous permet d'échanger des données entre le programme appelant « programme principal » et un programme appelé « sous programme ».

Passage de paramètres par valeur :

Exemple :
Etant donné une procédure qui calcule et affiche la somme de 2 variables. On veut lire « a » et « b » dans le programme principal et pas dans la procédure.

```
Procédure    Somme (val  x : entier, val  y : entier)
   Debut
      |          S : entier
      |          S = x + y
      |          Afficher (S)
   Fin
```

```
Debut
   |       a , b : entier
   |       Lire (a)
   |       Lire (b)
   |       Somme (a , b)
Fin
```

Remarque :
Les règles à respecter lors du passage des paramètres sont :

- Le nombre de paramètres ; exemple : si la procédure est défini avec 3 paramètres lors de l'appel on doit passer 3 paramètres.
- L'ordre de paramètres ; dans l'exemple précédent la valeur de a est affecté à x et la val de b est affecté à y.
- Le type de paramètres ; dans l'exemple précédent x est entier alors a doit être entier.

Exercice 1 :

Ecrire une procédure qui permet de produire deux entiers dont les valeurs sont liées dans le programme principal.

```
Procédure       Produit (val  a :entier , val  b :entier)
  | Debut
  |   |         Variables : P : entier
  |   |         P ← a * b
  |   |         Afficher (P)
  | Fin
Debut
  |           x , y  : entier
  |           Lire (x)
  |           Lire (y)
  |           Procuit (x , y)
 Fin
```

Exercice 2 :

Ecrire une procédure qui calcul le factoriel d'un entier avec l'entier saisis dans le programme principal.

```
Procédure       Factoriel (val  a : entier)
  | Debut
  |   |         Variables : F , i : entier
  |   |         F ← 1
  |   |         Pour i ←1  à   a   faire
  |   |               F←F* i
  |   |         FinPour
  |   |         Afficher (F)
  | Fin
Debut
  |           x  : entier
  |           Répéter
  |                 Lire (x)
  |           Jusqu'à (x>0)
  |           Factoriel (x)
 Fin
```

Exercice 3 :

Soit un tableau de taille 10 propose un algorithme qui permet de lire le contenu et affiche pour chaque case sa factoriel.

```
Procédure       Factoriel (val  a : entier)
  | Debut
  |   |
  |   |         Variables : F , i : entier
  |   |         F ← 1
  |   |         Pour i ←1  à   a   faire
  |   |               F←F* i
  |   |         FinPour
  |   |         Afficher (F)
  |  Fin
Debut
  |           Variables : j , tab[10]  : entier
  |           Pour j←0 à  9 faire
  |                 Répéter
  |                       Lire(tab[i])
  |                 Jusqu'à (tab[i] > 0)
  |           FinPour
  |           Pour j←0  à  9 faire
  |                  Afficher ("Le factoriel de ", tab[i] , " est : ")
  |                  Factoriel (tab[i])
  |           FinPour
 Fin
```

Exercice 4 :

Ecrire une procédure qui simule le fonctionnement d'une calculatrice avec les valeurs liées dans le programme principal.

```
Procédure       Calcul (val  x :entier , val  c :caractère , val  y : entier)
  | Début       Variables : R : réel
  |   |         Cas  c  parmi
  |   |               '+' : R←x + y
  |   |               '–' : R←x – y
  |   |               '/ ' : R←x  / y
  |   |               '* ': R←x * y
  |   |         FinCas
  |   |         Afficher (R)
  |  Fin
Debut
  |           Variables : a , b  : entier          S : caractère
  |           Lire (a)
  |           Répéter
  |                 Lire (S)
  |           Jusqu'à ( (S='+') ou (S='-') ou (S='*') ou (S='/') )
  |           Si (S = '/') alors
  |                 Répéter
  |                       Lire (b)
  |                 Jusqu'à (b <> 0)
  |           Sinon
  |                 Lire (b)
  |           FinSi
  |           Calcul (a , S , b)
 Fin
```

Passage de paramètre par variable :

On a l'algorithme suivant :

```
Procédure   Somme (val x : entier , val  y : entier)
  Debut
    |
    |       S : entier
    |       S ← x + y
    |       Afficher (S)
  Fin
Debut
    |       a , b , c : entier
    |       Lire (a)
    |       Lire (b)
    |       Somme ( a , b)
    |
 Fin
```

Cette fois ci on veut afficher le résultat de la somme dans le programme principal et pas dans la procédure.

Une solution pertinente est de passer les paramètres à la procédure et ceux par variable.

```
Procédure   Somme (val x : entier , val  y : entier , val  z : entier)
  Debut
    |
    |       z ← x + y
  Fin
Debut
    |       a , b , c : entier
    |       Lire (a)
    |       Lire (b)
    |       Somme ( a , b , c)
    |       Afficher (c)
    |
Fin
```

Exercice 1 :

Ecrire une procédure qui calcule le factoriel d'un entier saisi par l'utilisateur. L'affichage de variables se fait dans le programme principal.

```
Procédure   Factoriel ( val x : entier , val  F : entier)
  Debut
    |
    |       i : entier
    |       F ← 1
    |       Pour i←1  à x  faire
    |             F ← F * i
    |       FinPour
  Fin
Debut
    |       n , F : entier
    |       Répéter
    |             Lire (n)
    |       Jusqu'à (n>0)
    |       Factoriel ( n , Fact )
    |       Afficher (x, "! = " , Fact)
```

```
Fin
```

Exercice 2 :

Ecrire une procédure qui permet d'échanger les valeurs de 2 variables.
Résultat de l'affichage : - Avant Permutation a = 5 b = 6
- Après Permutation a = 6 b = 5

```
Procédure    Permutation (var x : entier, val  y : entier)
   Debut
      |      P : entier
      |      P ← x
      |      x ← y
      |      y ← P
   Fin
Debut
      |      a , b : entier
      |      Lire (a)
      |      Lire (b)
      |      Afficher ("Avant Permutation : a = ", a, "  b = ", b)
      |      Permutation (a , b)
      |      Afficher ("Après Permutation : a = ", a, "  b = ", b)
 Fin
```

Exercice 3 :

Ecrire une procédure qui simule le fonctionnement d'une calculatrice avec le résultat affiché dans le programme principal.

```
Procédure    Calcul (val x : entier, val  c : entier, val  y : entier, var s : réel)
   Debut
      |      Cas c parmi
      |            '+' : s ← x + y
      |            '–' : s ← x – y
      |            '*' : s ← x * y
      |            '/ ' : s ← x / y
      |      FinCas
   Fin
Debut        a, b : entier        r : réel
      |      signe : caractère
      |      Lire (a)
```

```
                Répéter
                    Lire (signe)
                Jusqu'à ((signe='+') ou (signe='-') ou (signe='*') ou (signe='/'))
                Si (signe = '/') alors
                    Répéter
                        Lire (b)
                    Jusqu'à (b<>0)
                Sinon
                    Lire (b)
                FinSi
                Calcul (a, signe, b, r)
                Afficher (r)
Fin
```

Exercice 4 :

Ecrire deux procédures pour remplir et affiche un tableau de 10 éléments.

```
Procédure   Remplir (var  x [ ] : entier)
   Debut
   |        i : entier
   |        Pour i←0  à  9  faire
   |            Lire (x [i])
   |        FinPour
   Fin
Procédure   Afficher (val x [ ] : entier)
   Debut
   |        i : entier
   |        Pour i←0  à  9  faire
   |            Afficher (x [i])
   |        FinPour
   Fin
Debut
   |        Tab [10] : entier
   |        Remplir (Tab)
   |        Afficher (Tab)
Fin
```

Exercice 5 :

Ecrire une procédure qui permet de chercher la valeur min et la valeur max dans un tableau. L'affichage de la valeur min, max et leur position se fait dans le programme principal.

```
Procédure   Recherche (val x [ ] :entier, var min1 :entier, var max1 :entier,
   Debut                  var  Posmin1 : entier, var  Posmax1 : entier)
                  i : entier
                  min1 ← x [0]
                  max1 ← x [0]
                  Posmin1 ← 0
                  Posmax1 ← 0
                  Pour i←1  à  9  faire
                        Si (x [i] <min1) alors
                              min1 ← x [i]
                              Posmin1 ← i
                         FinSi
                         Si (x [i] > max1) alors
                              max1 ← x [i]
                              Posmax1 ← i
                         FinSi
                   FinPour
    Fin
Debut
                  Tab [10], min, max, Posmin, Posmax : entier
                  Remplir (Tab, 10)
                  Recherche (Tab, min, max, Posmin, Posmax)
                  Afficher ("La val min est ", min," Sa position est : ", Posmin)
                  Afficher ("La val max est ", max, "Sa position est : ", Pasmax)
 Fin
```

Exercice 6 :

Ecrire une fonction qui simule le fonctionnement d'une calculatrice. Le résultat est affiché dans le programme principal.

```
Fonction    Calcul (val x : entier, val  y : entier, val z : caractère) : rée
   Debut
      |     R : réel
      |      Cas z parmi
      |            '+' : R ← x + y
      |            '–' : R ← x – y
      |            '*' : R ← x * y
      |            '/ ' : R ← x / y
      |      FinCas
      |      Retourne  R
   Fin
Debut
   |         a, b : entier
   |         T : réel
   |         s : caractère
   |         Lire (a)
   |         Répéter
   |               Lire (signe)
   |         Jusqu'à ((signe='+') ou (signe='-') ou (signe='*') ou (signe='/'))
   |         Si (signe = '/') alors
   |               Répéter
   |                     Lire (b)
   |               Jusqu'à (b<>0)
   |         Sinon
   |               Lire (b)
   |         FinSi
   |         T ← Calcul (a, b, s)
   |         Afficher (T)
Fin
```

Exercice 7 :

Ecrire une fonction qui vérifie si un mot est un palindrome. Le résultat est affiché dans le programme principal.

Exercice 8 :

Ecrire une procédure qui permet de trier un tableau saisi par l'utilisateur.
Ecrire une fonction qui cherche si un élément existe dans le tableau. Faire l'appel de la procédure et de la fonction dans le programme principal.

Ex 7 :

```
Fonction    Palindrome (val x [ ] : caractère, val  y : entier) : booléen
   Debut
        i, j : entier
        R : booléen
        i ← 0
        j ← y – 1
        R ← vrai
        TantQue ((R = vrai) et (i < j)) faire
                Si (x [i] = x [j]) alors
                        i ← i + 1
                        j ← j – 1
                Sinon
                        R ← faux
                FinSi
        FinTantQue
        Retourne  R
   Fin
Debut
        i , t [30] : entier
        i ← 0
        Répéter
                Lire (t [i])
        Jusqu'à (( t [i – 1] = "." ) ou ( i >=30))
        Si (Palindrome (t [ ] = vrai) alors
                Afficher ("Le mot est palindrome.")
        Sinon
                Afficher ("Le mot n'est pas palindrome.")
        FinSi
Fin
```

Ex 8 :

```
Procédure   Trier (val x [ ] : caractère, val  y : entier)
   Debut    i , j , Min , Pos: entier
            Pour i←0  à  n - 2  faire
                  Min ← Tab[i]
                  Pos ← i
            Pour j ←i+1  à  n - 1  faire
                  Si (Tab[j] < Min) alors
                        Min ← Tab[j]
                        Pos ← j
                  FinSi
            FinPour
            Si (Min <> tab [i]) alors
                ¨Pour j←Pos  à  i+1  (pas= -1) faire
                      Tab[j] ← Tab[j – 1]
                  FinPour
                  Tab[i] ← Min
            FinSi
    Fin
Fonction    Recherche (val  tab [ ] : entier, val  n : entier, val   v:entier) : booléen
   Debut    D , F , m : entier
            B : booléen
            D ← 0
            F ← n – 1
            B ← faux
            TantQue ((B = faux) et (D <= F)) faire
                  M ← (D + F) / 2
                  Si (v = tab [m]) alors
                        B ← vrai
                  Sinon
                        Si (v > tab[m]) alors
                              D ← m + 1
                        Sinon
                              F ← F – 1
                        FinSi
                  FinSi
            FinTantQue
    Fin
 Debut      tab [10] , v : entier
            B : booléen
            Remplir (tab)
            Trier (tab)
            Afficher ("Donner l'élément à rechercher : ")
            Lire (v)
            B ← Recherche (tab , 10 , v)
            Si (B = vrai) alors
                  Afficher (v, "existe dans le tableau.")
            Sinon
                  Afficher (v, "n'existe pas dans le tableau.")
         FinSi
Fin
```

5 – Les Structures : Enregistrement

Définition :

Les structures permettent à l'utilisateur de rassembler qu sein d'une seul entité un ensemble d'éléments qui ne sont pas nécessaires de même type.
Pour définir une structure, on utilise la syntaxe suivante : (champ ou bien attribut)

```
Type Structure Nom Structure
   | Nom champ 1 : Type
   | Nom champ 2 : type
   | …..
   | Nom champ n : Type
FinType
```

Remarque :

Le champ peut être de type simple ou lui-même de type structure.

Exemple :

Un stagiaire est identifié par son matricule, son nom, son prénom et sa moyenne.

```
Type Structure Stagiaire
   | Matricule : entier
   | Nom : chaîne de caractère
   | Prénom : chaîne de caractère
   | Moyenne : réel
FinType
```

On déclare une variable de type Stagiaire et pour accéder au champ de la variable on utilise une adresse par exemple :

```
Variables : Stag : Stagiaire
            Stag.Matricule ← 1
            Stag.Nom ← '' Amr ''
            Stag.Prénom ← '' Adibe ''
            Stag.Moyenne ← 14,5
```

Exercice 1 :

Ecrire une procédure qui permet de saisir un stagiaire.
Ecrire une procédure qui permet d'afficher un stagiaire.
Ecrire l'appel de ces procédures dans un programme principal qui permet de saisir et afficher 30 stagiaires.

```
Type   Structure   Stagiaire
   |       Matricule : entier
   |       Nom : chaîne de caractère
   |       Prénom : chaîne de caractère
   |       Moyenne : réel
FinType
Procédure    SaisirStag (var  Stag : Stagiaire)
   |  Debut
   |     |        Lire (Stag.Matricule)
   |     |        Lire (Stag.Nom)
   |     |        Lire (Stag.Prénom)
   |     |        Lire (Stag.Moyenne)
   |   Fin
Procédure     AfficherStag (val   Stag : Stagiaire)
   |  Debut
   |     |        Afficher ("Le matricule : ", Stag.Matricule)
   |     |        Afficher ("Le nom : ", Stag.Nom)
   |     |        Afficher ("Le prénom : ", Stag.Prénom)
   |     |        Afficher ("La moyenne :  ", Stag.Moyenne)
   |   Fin
Debut
   |             Variables : i : entier
   |                         TabStag [30] : Stagiaire
   |             Pour i←0  à  29  faire
   |                   Afficher ("Veuillez saisir les infos du stagiaire N°", i + 1)
   |                   SaisirStag (TabStag [i])
   |             FinPour
   |             Pour i←0  à  29  faire
   |                   AfficherStag (TabStag [i])
   |             FinPour
 Fin
```

Imbrication de structures :

Dans une structure on peut avoir des attributs qui sont de type Structure. Par exemple :

```
Type    Structure   Date
   |       Jour : entier
   |       Mois : entier
   |       Année : entier
FinType

Type    Structure   Stagiaire
   |       Matricule : entier
   |       Nom, Prénom : chaîne de caractères
   |       DateNaissance : Date
   |       Moyenne : réel
FinType
```

Exercice 2 :

Ecrire une fonction qui permet de lire une date.
Ecrire une procédure qui permet d'afficher une date sous le format suivant : ----/----/-------

```
Type   Structure   Date
   |       Jour, Mois, Année : entier
FinType
Fonction    LireDate (  ) : Date
    |  Debut
    |     |          Variables : D : Date
    |     |           Lire (D.Jour)
    |     |           Lire (D.Mois)
    |     |           Lire (D.Année)
    |     |           Retourne D
    |     Fin
Procédure     AfficherDate (val   D : Date)
     Debut
        |          Afficher (D.Jour , "/" , D.Mois , "/", D.Année)
      Fin
```

Exercice 3 :

Ecrire une fonction qui permet de saisir un stagiaire.
Ecrire une procédure qui permet d'afficher un stagiaire.
Ecrire une procédure qui permet de trier un tableau de 30 stagiaires par moyenne par ordre décroissant.
Faire l'appel de ces procédures et fonctions dans un programme principal qui permet de saisir 30 stagiaire , les trier et les afficher.

////// **IL FAUT LE CORRIGE DE L'EXERCICE J'AI PAS SAISI ENCORE**

6 - Les Fichiers :

Jusqu'à présent les informations utilisées dans nos programmes ne pourraient prévenir que de deux sources :

→ Soit, elles étaient incluses dans l'algorithme lui-même par le programmeur.

→ Soit, elles étaient entrées à la saisie par l'utilisateur.

Si on veut trouver les données saisies d'une exécution à une autre, on utilise **Les fichiers**.

Définition :

Un fichier est un ensemble de données (informations) stockées d'une façon permanente dans une mémoire de masse comme un disque dur, un cd, une clé usb.

Un fichier a deux types :

- **Un nom physique :** avec lequel il est connu par le système d'exploitation. Ex : stagiaire.txt
- **Un nom logique :** avec lequel il est connu par le programme.

Types de fichiers :

- **Fichier Texte :**
  - **+** Dans le fichier texte, on stocke l'information texte selon le code ASCII.
  - **+** Un fichier texte a l'extension (.Txt) donc il peut être lu par n'importe quel éditeur de teste.
  - + On peut stocker du teste (des caractères alphabétiques, spéciaux, des chiffres …)
  - + On utilise l'accès séquentiel.

- **Fichier Binaire :**
  - + Les données sont écrites comme en mémoire vive (des bits) donc l'accès du processeur est Rapide et la taille du fichier est réduite.
  - + Le fichier à l'apparence d'une suite d'octets illisibles.
  - + On peut stocker du texte mais aussi on peut stocker du son, image, un programme exécutable.
  - + On utilise l'accès séquentiel ou direct.

Mode D'accès :

Il y a principalement 2 types d'accès au données d'un fichier c'est-à-dire la manière dont la machine va pouvoir aller rechercher les informations contenus dans le fichier. On distingue :

- **Accès Séquentielle :**
  On ne peut accéder à une information qu'en ayant au préalable examiner celle qui le précède.

- **Accès Directe (Aléatoire) :**
  On peut accéder directement à l'enregistrement de son choix en précisant le N° de cette enreg.

Mode D'ouverture :

- **En Lecture :**
  - **+** Si le fichier existe, le compilateur se positionne sur le début du fichier : on peut lire les infos qu'il contient sans pouvoir les modifier.
  - **+** Si le fichier n'existe pas, le compilateur donne une erreur.

- **En Ecriture :**
  - + Si le fichier existe, le compilateur écrase le contenu et on pourra mettre les infos qu'on veut.
  - + Si le fichier n'existe pas, le compilateur le crée et on pourra mettre les infos qu'on veut.

- **En Ajout :**
  + Dans ce mode il n'y a pas de distraction des fichiers. Le pointeur se positionne à la fin du fichier pour l'écriture.

Manipulation du fichier :

1) Déclaration du Fichier :

F : Fichier

2)Lien entre le physique et logique :

Lien (nomphysique , nomlogique)

3) Ouvrir le fichier :

Ouvrir (nomlogique , typefichier , moded'accès , moded'ouverture)

4) Lire (var)

5) EcrireFichier (nomlogique , var)

6) FermerFichier (nomlogique)

Exercice 1 :

Ecrire dans un fichier

```
Debut
        Variables : F : Fichier          c : caractère
        Lire ( "c : / data . txt ", F)
        Ouvrir (F , "t" , "seq" , "écriture")
        Répéter
            Afficher ("Saisir un caractère : ")
            Lire (c)
            EcrireFichier(F , c)
         Jusqu'à (c = '#')
         FermerFichier(F)
Fin
```

Exercice 2 :

Lire un Fichier

```
Debut
        Variables : F : Fichier          c : caractère
        Lire ( "c : / data . txt ", F)
        Ouvrir (F , "t" , "seq" , "Lecture")
        Répéter
            LireFichier(F,c)
            Afficher (c)
         Jusqu'à EOF (F)                 EOF : end of file
         FermerFichier(F)
Fin
```

Exercice 3 :

Lire un Fichier

```
Procédure   Ajouter (F : Fichier , val S : Stagiaire)
  Début
            Ouvrir (F , "b" , "seq" , "Ajout")
            EcrireFichier (F , s)
  Fin

Procédure   Lecture (F : Fichier , val t[ ] : Stagiaire , var N : entier)
  Début
            Variables : c : entier
            Ouvrir (F , "b" , "seq" , "Lecture")
            C ← 0
            Repeter
                  LireFichier (F , t[c])
                  C ← C+1
            Jusqu'à  EOF (F)
            N ← C – 1
            FermerFichier(F)
   Fin

Debut
            Variables :TabStag[100] , Stag : Stagiaire
                        F : Fichier          rep : caractère          i , Nb : entier
            Lire ( "c : / data . txt ", F)
            Répéter
                  SaisirStag (Stag)       // Procédure de saisie Stag
                  Ajouter (F , Stag)
                  Ouvrir ("Voulez vous saisir un autre stagiaire O/N")
                  Lire(rep)
            Jusqu'à (rep = 'N' ou rep = 'n')
            Lecture (F , TabStag , Nb)
            Pour i ← 0  à  Nb-1  faire
                  AfficherStag (TabStag[i])
            FinPour
            FermerFichier(F)
Fin
```