Procédures & Fonctions

PROBLÈME

- Comment réutiliser un algorithme existant sans le réécrire ?
- Comment structurer un algorithme pour le rendre plus compréhensible ?

SOLUTION

- Décomposer l'algorithme en sous-algorithmes (Fonctions et Procédures)
- Trouver une solution à chacun puis regrouper le tout dans un seul algorithme

FONCTIONS

❖ Définition

- une fonction est un sous-programme destiné à effectuer un enchaînement de traitements à l'aide d'un simple appel.

- une fonction a pour but principal d'effectuer un calcul puis de renvoyer un (**et un SEUL**) résultat.

FONCTIONS

❖ Syntaxe d'une fonction

En algorithme

Fonction Nom_fonction (liste des paramètres : type**) : type**

Var

{liste des variables locales}

Début

<Séquence d'instructions>

Nom_fonction← Résultat {Obligatoire}

Fin Fonction

En VBA

Function Nom_fonction (paramètre1 **As** type, Paramètre2 **As** type,……**)** **As type**

instruction1

instruction2

.....

Nom_fonction = Résultat

End Function

FONCTIONS

❖ Remarques

- Le type de la fonction doit obligatoirement conclure la ligne d'en-tête, après la liste de paramètres
- L'ordre, le type et le nombre des arguments doivent être respectés lors de l'appel de la fonction.
- Une fonction retourne toujours une valeur
- Une fonction Nom_fonction contient toujours une instruction de la forme

  Nom_fonction ← Expression

- En général, l'utilisation d'une fonction se fait, dans le corps du programme principal
  - ✓ Soit par une affectation: v ← **Nom_fonction** (paramètres effectifs)
  - ✓ Soit dans l'écriture: Ecrire (**Nom_fonction** (paramètres effectifs))

FONCTIONS

❖ Exemple

Fonction qui retourne le carré d'un entier

N=3 ⟶ carré(N)=9

```
Fonction carré(n : entier): entier
  Début
     carré ← n * n
fin Fonction
```

Utilisation dans un algorithme

```
Algorithme exemple1
Variable i, j: entier
Fonction carré(n : entier): entier
  Début
        carré ← n * n
  fin fonction
Début
  Lire (i)
  j ← carré(i)
   Ecrire(j)    /* Ecrire(carré(i))*/
Fin
```

Paramètre formel

Paramètre effectif

FONCTIONS

❖ Exercice

1. Ecrire une fonction qui
   - ✓ Prend un tableau de 10 entiers, puis
   - ✓ Retourne la valeur Vraie ou Faux selon que le tableau est trié par ordre croissant ou non

Démarche

On suppose d'abord que le tableau est trié

Ensuite on compare chaque case à sa suivante:

Si l'ordre n'est pas respecté alors on conclut que le tableau n'est pas trié

FONCTIONS

❖ Correction

Fonction Trié(T: Tableau[10] entiers): booléen

Variables
i : entier
b : booléen

Début
b ← Vrai
Pour i de 1 à 9
Si T(i) > T(i+1) alors
b ← Faux
FinSi
FinPour

Trié ← b

Fin Fonction

2. Ecrire un algorithme qui
   - lit un tableau de 10 entiers puis
   - teste s'il est trié ou pas
3. Traduire le code VBA

FONCTIONS

```
Algorithme exercice
   Variables
      T1 : Tableau[10]  enties
       i : entier
Fonction Trié( T: Tableau[10] entiers): booléen
      Variables
        i : entier
           b : booléen
      Début
            b ← Vrai
            Pour i = 1 à 9
                  Si T(i) > T(i+1) alors
                        b ← Faux
                  FinSi
            FinPour
            Trié ← b
   Fin Fonction
```

Paramètre formel

```
Début
  Pour i  de 1 à 10 faire
        Lire( T1[i])
  Fin Pour
  Si Trié(T1) = Vrai Alors
    Ecrire(''c'est trié'')
  Sinon
     Ecrire(''Non trié'')
FinSi
Fin
```

Paramètre effectif

FONCTIONS

Code VBA

```
Function Trié(t() As Integer) As Boolean
    Dim b As Boolean
    Dim i As Integer
     b = True
     For i = 1 To 4
        If t(i) > t(i + 1) Then
              b = False
        End If
     Next
     Trié = b
End Function
```

```
Sub exercice()
   Dim T1(10) As Integer
   Dim i As Integer
   For i = 1 To 5
        T1(i) = InputBox("donner T[" & i & " ]")
    Next
    If Trié(T1) = True Then
          MsgBox ("trié")
    Else
          MsgBox ("non trié")
     End If
End Sub
```

PROCÉDURES

❖ Définition

- Une procédure est un bloc d'instructions nommé et déclaré dans l'entête de l'algorithme et appelé dans son corps à chaque fois que le programmeur en a besoin

- Une procédure est une fonction qui ne renvoie pas de résultat

PROCÉDURES

❖ Syntaxe d'une Procédure

En algorithme

Procédure **Nom Procedure (**liste des paramètres : type**)**

Var

{liste des variables locales}

Début

<Séquence d'instructions>

Fin Procédure

En VBA

Sub **Nom Procédure (**paramètre1 **As** type, Paramètre2 **As** type,….**)**

instruction1

instruction2

.....

End Sub

PROCÉDURES

❖ Remarques

- lorsque la déclaration ci-dessus est faite, il suffit ensuite d'écrire le nom de la procédure dans le bloc principal pour déclencher la liste des actions décrites.
- En VBA pour appeler une procédure , il faut utiliser le mot clé Call:

Call nom_procédure(param1,param2,…)

- Une procédure peut appeler d'autres sous-programmes définis avant elle.
- En général, une procédure modifie la valeur de ses paramètres (passage par adresse ou par référence)

PROCÉDURES

❖ Exemple

Une procédure qui ajoute 2 à un entier

Procédure aug2(Var n : entier)

Début

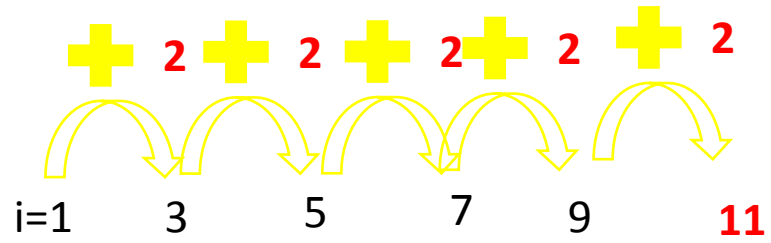
n ← n+2

Fin Procédure

Utilisation dans un algorithme

Ecrire un algorithme qui

- Lit un entier positif n puis
- Affiche tous les nombres impaires inférieurs à n



N=10

Condition d'arret : i>N

```
Algorithme ex1
  Variables  i,n: entier
 Procédure aug2(Var n : entier)
  Début
  n ← n+2
 Fin Procédure
 Début
    Répéter Lire(n) jusqu'à (n>=1)
    i ←1
     Tant que i <n
        Ecrire(i)
        aug2(i)
    Fin TantQue
 Fin
```

PROCÉDURES

Algorithme ex1

Variables i,n: entier

Procédure aug2(Var n : entier)

Début

n ← n+2

Fin Procédure

Début

Répéter Lire(n) jusqu'à (n>=1)

i ←1

Tant que i <n

Ecrire(i)

aug2(i)

Fin TantQue

Fin

Code VBA

```
Sub aug2(n As Integer)
        n = n +2
End sub
```

```
Sub ex1()
Dim i, n As Integer
Do  n=InputBox(″donner n″) Loop until (n>=1)
i = 1
While (i <n)
   MsgBox(i)
   Call aug2(i)
Wend
End Sub
```

PASSAGE DES PARAMÈTRES

❖ Passage de paramètres par valeur

- Passer un paramètre par valeur revient à n'utiliser que la valeur de la variable au moment où elle est passée en paramètre.
- **À la fin de l'exécution du sous- programme, la variable conservera sa valeur initiale.**
- Dans ce cas, la procédure (ou fonction) travaille sur une copie des paramètres.
- C'est le mode de transmission par défaut en algorithme (Non pas pour VBA).
- **Syntaxe algorithmique**: PROCEDURE <nom_procédure> (Val param1 :type1 , Val param2, param3 :type2)
- **Syntaxe VBA**: Sub Nom Procédure (ByVal paramètre1 As type, ByVal Paramètre2 As type,….)

Optionnel

obligatoire

PASSAGE DES PARAMÈTRES

Exemple : Soit l'algorithme suivant.

Algorithme passage-val

Variable M : entier

Procédure P1 (nombre : entier)

Début

nombre ← 2*nombre

Ecrire (nombre)

Fin

Début

Lire (M)

P1 (M)

Ecrire (M)

Fin

Exécutons cet algorithme pour la valeur (10)

1. Avant l'appel de procédure : la seule variable déclarée est la variable globale (M) : M=10
2. Après l'appel de procédure : la variable-paramètre "nombre" est déclarée et reçoit en copie la valeur de M.: Nombre=10
3. Apres l'exécution de la procédure : Nombre=20
4. Au retour à l'algorithme (au niveau de l'appel) il ne reste que la variable globale avec sa valeur initiale : M=10

Code VBA:

```
Sub P1(ByVal nombre As Integer)
nombre = nombre * 2
MsgBox ("nombre = " & nombre)
End Sub
```

```
Sub paasege_valeur()

Dim M As Integer
M = InputBox("donner un entier")
Call P1(M)
MsgBox ("M= " & M)

End Sub
```

PASSAGE DES PARAMÈTRES

❖ Passage de paramètres par adresse (ou par variable)

- Ici, il s'agit non plus d'utiliser simplement la valeur de la variable, mais également son emplacement dans la mémoire (d'où l'expression « par adresse »).

- En fait, le paramètre formel se substitue au paramètre effectif durant le temps d'exécution du sous-programme. **Et à la sortie il lui transmet sa nouvelle valeur**.

- **Syntaxe algorithmique**: PROCEDURE <nom_procédure> (Var param1 :type1 ; Var param2, param3 :type2)

obligatoire

- **Syntaxe VBA**: Sub Nom Procédure (ByRef paramètre1 As type, ByRef Paramètre2 As type,….)

Optionnel

- *NB*: en VBA , quand la façon de passer les arguments n'est pas précisée, l'argument est passé par référence (ByRef).

En clair, le passage ByRef est le passage par défaut en VBA.

PASSAGE DES PARAMÈTRES

Exemple : Soit l'algorithme suivant.

Algorithme passage-val

Variable M : entier

Procédure P1 (Var nombre : entier)

Début

nombre ← 2*nombre

Ecrire (nombre)

Fin

Début

Lire (M)

P1 (M)

Ecrire (M)

Fin

Exécutons cet algorithme pour la valeur (10)

1. Avant l'appel de procédure : la seule variable déclarée est la variable globale (M) : M=10
2. Après l'appel de procédure : la variable-paramètre « nombre » se substitue à la variable M: Nombre=10
3. Au retour à l'algorithme il ne reste que la variable globale avec sa nouvelle valeur. M=20

Code VBA:

```
Sub P1(nombre As Integer)
nombre = nombre * 2
MsgBox ("nombre = " & nombre)
End Sub
```

```
Sub paasege_valeur()

Dim M As Integer
M = InputBox("donner un entier")
Call P1(M)
MsgBox ("M= " & M)

End Sub
```

EXERCICES

1. Écrire un programme qui permet de :
   - lire deux entier a et b,
   - permuter le contenu de ces variables en se servant d'un sous-programme (procédure ou fonction).

Algorithme exercice 1
Variables
A, b : entiers
Procédure Permuter (Var x: entier, Var y: entier)
Variables Aux: entier
Début
Aux←x
x ←y
y ←Aux
Fin Procédure
Début
Ecrire(''Donner deux entiers'')
Lire (a) lire(b)
Permuter(a, b)
Ecrire('' après permutation a= '' , a, '' b='' , b)
Fin

```
Sub permut(x As Integer, y As Integer)
Dim Aux As Integer
Aux = x
x = y
y = Aux
End Sub
```

```
Sub exercice()

Dim a As Integer
Dim b As Integer
a = InputBox("donner a")
b = InputBox("donner b")
Call permut(a, b)
MsgBox ("apres permutation a = " & a & " b= " & b)

End Sub
```

EXERCICES

2. Ecrire un sous programme **NombreChiffre** permettant de fournir le nombre de chiffres d'un entier N.

   Ecrire un petit programme qui teste le sous programme NombreChiffre

Exemple d'exécution:

Introduire un nombre: 123 ⟶ Le nombre 123 a 3 chiffres.

Introduire un nombre: 580499 ⟶ Le nombre 580499 a 6 chiffres.

```
Algorithme exercice2
Variables N, nb: entier
Fonction NombreChiffre (Val X: entier):
entier
   Variables i : entier
   Début
     i=0
     Tant que (X<>0) faire
       X←X div 10
       i ← i+1
     Fin Tant que
   NombreChiffre ←i
Fin Fonction
```

```
Début
   écrire(''donner un entier'')
    Lire(N)
     nb ←NombreChiffre (N)
     écrire(''Le nombre de chiffres de'', N, ''='',nb)
/* écrire(''Le nombre de chiffres de'', N, ''='', NombreChiffre (N))
Fin
```

EXERCICES

Algorithme exercice2
Variables N, nb: entier
Fonction NombreChiffre (Val X: entier): entier
Variables i : entier
Début
i=0
Tant que (N<>0) faire
N←N div 10
i ← i+1
Fin Tant que
NombreChiffre ←i
Fin Fonction
Début
écrire(''donner un entier'')
Lire(N)
nb ←*NombreChiffre (N)*
écrire(''Le nombre de chiffres de'', N, ''='',nb)
/ écrire(''Le nombre de chiffres de'', N, ''='', NombreChiffre (N))*
Fin

```
Function NombreChiffre(ByVal X As Double) As Integer
    Dim i As Integer
      i = 0
    While (X <> 0)
        X = X \ 10
        i = i + 1
     Wend
  NombreChiffre = i
End Function
```

```
Sub exercice2()
Dim N As Double
Dim nb As Integer
N = InputBox("donner N")
nb = NombreChiffre(N)
MsgBox ("le nombre de chiffre de " & N & " = " & nb)
End Sub
```