Plus loin avec la programmation…

STRUCTURES ALGORITHMIQUES

Permet d'activer une partie du code en fonction de la réalisation d'une condition ou pas.

Syntaxe

```
If condition Then
  bloc d'instructions
  si la condition est vraie
Else
  bloc d'instructions
  si la condition est fausse
End If
```

(1) Condition est souvent une opération de comparaison

(2) La valeur de retour de Condition est de type booléen (True ou False)

(3) Then doit être sur la même ligne que If

(4) La partie Else est facultative (ne rien faire si la condition est fausse)

(5) Il est possible d'imbriquer une autre structure conditionnelle If dans les blocs d'instructions

Branchement conditionnel IF – Un exemple

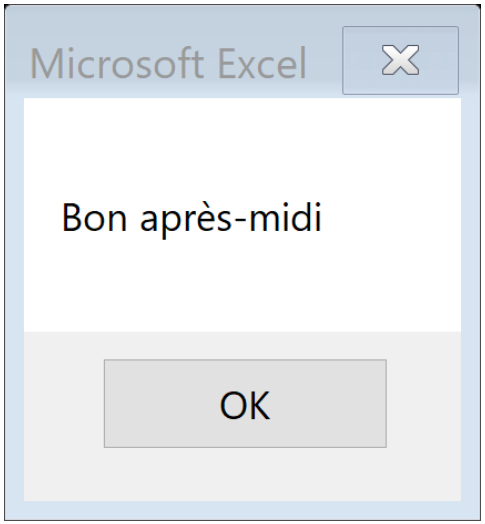
Entrées : prix HT (réel), catégorie de produit (chaîne)
Sortie : prix TTC (réel)

```
Public Function MonTTCBis(pht As Double, cat As String) As Double
'déclarer la variable de calcul
Dim pttc As Double
'en fonction de la catégorie de produit
If (cat = "luxe") Then
    pttc = pht * 1.33
Else
    ‘la valeur de cat est différente de ‘’luxe’’
    pttc = pht * 1.2
End If
'renvoyer le résultat
MonTTCBis = pttc
End Function
```

Branchement conditionnel IF/ElseIf – Un exemple

```
Sub bonjour()
Dim msg As String
If Time < 0.5 Then
    msg = "jour"
ElseIf Time < 0.75 Then
    msg = "après-midi"
Else
    msg = "soir"
End If
MsgBox "Bon" & msg

End Sub
```



Permet d'activer une partie du code en fonction des valeurs prises par une variable de contrôle. Peut se substituer au IF, mais pas toujours, tout dépend de la forme de la condition (*condition composée, on doit passer par un IF*).

Syntaxe

```
Select Case variable
        Case valeur 1
                bloc d'instructions
        Case valeur 2
                bloc d'instructions
        ...
        Case Else
                bloc d'instructions
End Select
```

(1) Variable est la variable de contrôle, elle peut être de n'importe quel type en VBA, y compris un réel ou une chaîne de caractères

(2) Valeur doit être de type compatible avec variable

(3) La partie Case Else est facultative

(4) L'imbrication avec un autre IF ou un autre Select Case (autre variable de contrôle) est possible.

Branchement multiple SELECT CASE – Un exemple

Entrées : prix HT (réel), catégorie de produit (chaîne)
Sortie : prix TTC (réel)

```
'fonction select case
Public Function MonTTCSelon(pht As Double, cat As String) As Double
'déclarer la variable de calcul
Dim pttc As Double
'en fonction de la catégorie de produit
Select Case cat
    Case "luxe"
        pttc = pht * 1.33
    Case Else
        pttc = pht * 1.2 'toute autre valeur que ‘’luxe’’
End Select
'renvoyer le résultat
MonTTCSelon = pttc
End Function
```

Il est possible d'introduire des plages de valeurs dans la partie Case de la structure Select Case. La comparaison devient plus sophistiquée. Variable est un numérique dans ce cas, entier ou même réel.

Syntaxe

```
Select Case variable
       Case Is op.de.comparaison valeur
              bloc d'instructions

       Case valeur de départ To valeur de fin
              bloc d'instructions

       Case Else
              bloc d'instructions
End Select
```

Branchement multiple SELECT CASE – Plages de valeurs – Un exemple

Entrée : quantité (entier)
Sortie : prix unitaire (réel)
Calcul : quantité < 100 → p.u. = 0.5
100 ≤ quantité ≤ 200 → p.u. = 0.3
quantité > 200 → p.u. = 0.2

```
'calcul du prix unitaire en fonction de la quantité
Public Function MonPU(quantite As Long) As Double
'variable intermédiaire
Dim pu As Double
'selon les valeurs de quantité
Select Case quantite
    Case Is < 100
        pu = 0.5
    Case 100 To 200
        pu = 0.3
    Case Is > 200 'Case Else aurait fait l'affaire aussi
        pu = 0.2
End Select
MonPU = pu
End Function
```

Faire répéter l'exécution d'un bloc d'instructions. Le nombre d'itérations est contrôlé par un indice.

Syntaxe

```
For indice = val.départ to val.fin step pas
        bloc d'instructions
        ...
Next indice
```

(1) Indice est un type ordonné, très souvent un numérique

(2) pas contrôle le passage d'une valeur à l'autre d'indice, si omis, pas = 1 par défaut

(3) Next entérine le passage à la valeur suivante de indice, si cette prochaine valeur est > à val.fin, on sort de la boucle

(4) Val.fin doit être superieure à val.départ pour que l'on rentre dans la boucle

(5) Si pas est négatif, val.fin doit être inférieure à val.départ cette fois-ci

(6) L'instruction Exit For permet de sortir prématurément de la boucle

(7) On peut imbriquer des boucles (une boucle à l'intérieur d'une autre boucle)

Entrée : n (entier)
Sortie : S (réel)
Calcul : $S = 1^2 + 2^2 + \ldots + n^2$

```
'calcul de la somme des carrés des valeurs
Public Function MaSommeCarre(n As Long) As Double
'variables de calcul (s pour la somme, i : indice)
Dim s As Double, i As Long
'initialisation
s = 0
'boucle avec l'indice i
For i = 1 To n Step 1
    s = s + i ^ 2
‘Next joue le rôle de l’incrémentation (i suivant)
Next i
'renvoyer le résultat
MaSommeCarre = s
End Function
```

Faire répéter l'exécution d'un bloc d'instructions. Le nombre d'itérations est contrôlé par une condition. Attention à la boucle infinie c.-à-d. la condition permettant de sortir de la boucle n'est jamais déclenchée.

Syntaxe

```
Do While condition
        Bloc d'instructions...
        ...
Loop
```

(1) Condition est un booléen, c'est souvent une opération de comparaison

(2) On continue l'exécution TANT QUE la condition est vraie ; si la condition est fausse, on sort de la boucle

(3) Exit Do permet de provoquer la sortie prématurée de la boucle

Si la condition est fausse d'emblée. On peut ne pas rentrer dans la boucle.

Boucle DO WHILE…LOOP (un exemple)

Entrée : n (entier)
Sortie : S (réel)
Calcul : $S = 1^2 + 2^2 + \ldots + n^2$

```
'calcul de la somme des carrés des valeurs
Public Function MaSommeCarreWhile(n As Long) As Double
'variables de calcul
Dim s As Double, i As Long
'initialisation
s = 0
'il nous revient aussi d'initialiser l'indice
i = 1
'boucle TANT QUE
Do While (i <= n)
    'sommer
    s = s + i ^ 2
    'pas de next, nous devons incrémenter l'indice
    i = i + 1
Loop
'renvoyer le résultat
MaSommeCarreWhile = s
End Function
```

Faire répéter l'exécution d'un bloc d'instructions. Le nombre d'itérations est contrôlé par une condition.

Syntaxe

```
Do
        Bloc d'instructions
        ...
        ...
Loop While condition
```

On est sûr de rentrer au moins une fois dans la boucle.

Le choix de la bonne structure (Faire.. Tant Que ou Tant Que.. Faire) dépend du problème à traiter

Les boucles DO contrôlées par une condition sont très riches en VBA.

```
Do { While | Until } condition
        [ statements ]
        [ Exit Do ]
        [ statements ]
Loop
-or-
Do
        [ statements ]
        [ Exit Do ]
        [ statements ]
Loop { While | Until } condition
```

Le Répeter… Jusqu'à (Until) existe aussi.