PROCÉDURES EN VBA

Le VBA permet de gérer plusieurs types de variable ainsi que des procédures appelant elles-mêmes des sous-procédures avec récupération de certaines valeurs. Il est également possible de gérer la position d'un classeur dans l'arborescence de Windows.

GESTION DES CELLULES ET DES CLASSEURS

GESTION DES CELLULES

RÉCUPÉRATION DE VALEURS D'UNE AUTRE FEUILLE

Stocke dans les variables *Montant*, *Remise* le contenu des cellules B18 et B19 situées dans la feuille « Gestion » :

```
With Sheets("Gestion")
    Montant = .[B18]
    Remise = .[B19]
End With
```

✓ Pour les coordonnées des cellules, il est également possible d'écrire `.Cells(i,1)`, `.Range("B2:B20")`, etc.

RECHERCHE D'UNE VARIABLE DANS UNE ZONE DE CELLULES

```
Dim RgRecherche As Range ' RgRecherche défini en tant que cellule (Range)
Set RgRecherche = Range("A1:C1000").Find("Foubert", Range("A1"))
If Not RgRecherche Is Nothing Then
    MsgBox RgRecherche.Row
End If
```

Recherche dans la zone de cellules A1:C2000 le nom *Foubert* en commençant par la cellule A1 et stocke la réponse dans la variable *RgRecherche*.

Si ce nom est trouvé — *RgRecherche* n'est pas vide (Not… Is Nothing) — alors, le numéro de la ligne contenant la réponse est affichée.

GESTION DE L'ARBORESCENCE D'UN CLASSEUR

ENREGISTREMENT D'UN CLASSEUR DANS UN DOSSIER ET UN NOM DE FICHIER

```
Dim ActuFichier As String
ActuFichier = ThisWorkbook.Path & "\Suivi\Gestion.xlsx"
ThisWorkbook.SaveAs ActuFichier
```

ENREGISTREMENT D'UN CLASSEUR AUTOMATIQUEMENT

```
ActiveWorkbook.Save
```

OUVERTURE D'UN CLASSEUR EN LIGNE

Exemple : récupération d'un tableau en ligne réalisé sur le site de Jotform :

```
Workbooks.Open ("http://eu.jotform.com/excel/8779906063")
```

GESTION DES DOSSIERS

ACCÈS À UN DOSSIER PARTICULIER

```
ChDir ("C:\Users\Foubert\Documents\Messagerie")
```

Accès à un sous-dossier du dossier actuel

```
ChDir (ThisWorkbook.Path & "\Moranges")
```

Accès au dossier précédent le dossier courant

```
ChDir ".."
```

Affichage du dossier contenant le classeur ouvert

```
MsgBox ThisWorkbook.Path
```

Création d'un dossier si celui-ci n'existe pas

Création du dossier « Messagerie » si celui-ci n'existe pas

```
Dim ActuDossier As String
ActuDossier = ThisWorkbook.Path & "\Messagerie"
' Création du dossier si celui-ci n'existe pas
If Dir(ActuDossier, vbDirectory) = "" Then
    MkDir (ActuDossier)
End If
' Accès au dossier
ChDir (ActuDossier)
```

Ouverture d'un lien hypertexte contenu dans une cellule

Ouvre un lien hypertexte contenu dans la cellule active :

```
ActiveWorkbook.FollowHyperlink Address:="http://" & ActiveCell, NewWindow:=True
```

Utilisation des variables

Variables privées et publiques

Les variables privées ne sont utilisables que dans la procédure en cours d'exécution alors que les variables publiques peuvent être accessibles de n'importe où dans le classeur.

Pour définir une variable accessible que dans une procédure

Placer la définition de la variable à l'intérieur de la procédure, exemple :

```
Sub Essor()
    Dim Adresse As Integer
    Adresse = 25
    Suite du code
End Sub
```

Pour définir une variable accessible à partir de toutes les procédures d'un module

Définir la variable avant toutes les procédures, exemple :

```
Dim Adresse As String
' Le contenu de Adresse sera accessible
' à partir de toutes les procédures de ce module uniquement
Sub Essor()
    Adresse = 25
    Envoi ' Appel de la macro Envoi dans le même module
End Sub
```

✓ `Private Adresse As String` est équivalent à `Dim` et montre bien que la variable *Adresse* est réservée à ce module.

POUR DÉFINIR UNE VARIABLE ACCESSIBLE À PARTIR DE TOUS LES MODULES D'UN CLASSEUR

Définir la variable avant toutes les procédures avec *Public* et non *Dim*, exemple :

```
Public Adresse As String
Sub Essor()
    Adresse = 25
    Call Module2.Suite ' Appel la macro Suite dans le module Module2
End Sub
```

POUR RÉCUPÉRER UNE VARIABLE DANS UN AUTRE CLASSEUR OUVERT :

Cela nécessite l'utilisation d'une sous-procédure dans le deuxième classeur.

Procédure dans le premier classeur (macro à exécuter) :

```
Public Adresse As String
Sub Essor()
    Adresse = 25
    ' Envoi du contenu de la variable Adresse et
    ' appel de la macro Import dans le module Module1 du classeur Classeur1.xlsm
    Application.Run "Classeur1.xlsm!Module1.Import", Adresse
End Sub
```

Sous-procédure dans le second classeur :

```
Sub Import(ByVal Afficher As String)
    MsgBox "Valeur de la variable du premier classeur : " & Afficher
End Sub
```

✓ La variable Afficher va recevoir le contenu de la variable (ici Adresse) mentionnée à la fin de l'appel de la procédure

POUR VIDER LE CONTENU D'UNE VARIABLE PRIVÉE OU PUBLIQUE

```
Adresse = Empty ' La variable Adresse perd son contenu
```

DÉFINIR UNE VARIABLE CONSTANTE POUR TOUTES LES MACROS DU MODULE

Une variable constante, dont la valeur est fixée au début du module, permet de rendre cette valeur accessible dans toutes les procédures du module.

Ici dans les macros *Groupe1*, *Groupe2* ou *Groupe3*, la variable *Taux* aura toujours comme valeur 0,1.

```
Const Taux As Single = 0.1
Sub Groupe1()
    MsgBox 10 * Taux
End Sub
Sub Groupe2()
    MsgBox 20 * Taux
End Sub
Sub Groupe3()
    MsgBox 30 * Taux
End Sub
```

✓ `Public Const` rend la variable accessible dans tous les modules du classeur.

VARIABLE EN TABLEAU

Il est possible de stocker plusieurs valeurs dans une même variable — sous forme de tableau à plusieurs dimensions — dans une seule variable ayant le même nom mais des numéros différents.

Exemple qui stocke dans une variable appelée *Contenu* la valeur des cellules de la ligne 1 à la ligne 5 et de la colonne 1 à la colonne 7 pour remettre ces valeurs dans une seconde feuille :

```
Sub Stockage()
    ' Dimensionne la variable contenu de 1 à 5 pour la première dimension
    ' et de 1 à 7 pour la seconde dimension
    Dim i As Integer, j As Integer
```

```
    Dim Contenu(1 To 5, 1 To 7) As String
    ' Ubound : compte jusqu'à la limite choisie (dimension 1 ou 2) de Contenu
    For i = 1 To UBound(Contenu, 1)
        For j = 1 To UBound(Contenu, 2)
            Contenu(i, j) = Cells(i, j)
            Worksheets("feuil2").Cells(i, j) = Contenu(i, j)
        Next j
    Next i
End Sub
```

CHRONOMÉTRAGE DE LA DURÉE D'UNE MACRO

```
    Dim Chrono As Date
    ' Début du chronométrage de la durée de la macro
    Chrono = Now()
    Code de la macro...
    ' Fin du chronométrage de la durée de la macro
    MsgBox "La macro a duré " & Chr(10) & Format(Now() - Chrono, "hh:mm:ss")
```

COPIER-COLLER DE CELLULES

EN PASSANT PAR UN COPIER-COLLER

Copie le contenu des cellules de la 1ère à la 8e colonne de la ligne courante après la dernière cellule non vide de la colonne A

```
Sub Copie()
    Range(Cells(ActiveCell.Row, 1), Cells(ActiveCell.Row, 8)).Select
    Selection.Copy ' Copie les cellules sélectionnées
    ' Sélection de la cellule après la dernière cellule non vide de la colonne A
    Range("A2").End(xlDown).Offset(1, 0).Select
    ActiveSheet.Paste' Colle les cellules copiées dans la dernière cellule
    ActiveCell.Offset(0, 1).Select
End Sub
```

✓ Pour ne copier-coller que les valeurs (résultats des calculs) sans la mise en forme, remplacer *ActiveSheet.Paste* par *Selection.PasteSpecial Paste:=xlPasteValues*

EN PASSANT PAR UNE VARIABLE (MEILLEURE MÉTHODE)

Copie le contenu des cellules de la 1ère à la 8e colonne de la ligne courante vers la cellule A30 en stockant leur contenu dans la variable ZoneOrigine

```
Sub CopieVariable()
    Dim ZoneOrigine As Range
    Set ZoneOrigine = Range(Cells(ActiveCell.Row, 1), Cells(ActiveCell.Row, 8))
    ' Sélection de la cellule après la dernière cellule non vide de la colonne A
    Range("A2").End(xlDown).Offset(1, 0).Select
    ' Copie le contenu de la variable ZoneOrigine à partir de la cellule A30 :
    ZoneOrigine.Copy Destination:=ActiveCell
End Sub
```

COPIE EN UNE SEULE LIGNE

```
Worksheets("Nord").Range("B5:C9").Copy Worksheets("Suivi").Range("A6")
```

Récupère le contenu des cellules B5 à C9 de la feuille *Nord* pour les copier à partir de la cellule A6 de la feuille *Suivi*

COPIER-COLLER D'UN CLASSEUR À UN AUTRE

```
Workbooks("Ventes.xlsx").Worksheets("Nord").Range("B5:C9").Copy _
  ThisWorkbook.Worksheets("Suivi").Range("B2")
```

Copie les cellules de B5 à C9 de la feuille *Nord* du classeur *Venets.xlsx* vers la cellule B2 de la feuille *Suivi* du classeur à partir duquel a été lancé la macro

TEXTE EN MAJUSCULES OU EN MINUSCULES

Voici quelques fonctions pour passer des minuscules aux majuscules ou ne mettre que les premières lettres en majuscule :

```
Sub MajMin()
    Dim ActuTexte as String
    ActuTexte = "BONJOUR à tous"
    MsgBox UCase(ActuTexte) ' Majuscule
    MsgBox LCase(ActuTexte) ' Minuscule
    MsgBox Application.Proper(ActuTexte) ' Premières lettres en majuscule
End Sub
```

PROCÉDURES ET SOUS-PROCÉDURES

PROCÉDURES PUBLIQUES ET PRIVÉES

Une procédure peut appeler une sous-procédure. L'intérêt d'une sous-procédure est que c'est un code unique qui peut être appelée de plusieurs procédures afin d'effectuer des calculs, modifier des variables, etc.

Les procédures publiques sont accessibles de l'ensemble des modules alors que les procédures privées ne sont accessibles que du module en cours. Ainsi, si la procédure est privée, il pourra y avoir plusieurs procédures du même nom mais avec des codes différents dans des modules différents.

- La procédure publique commence par *Public* (qui est facultatif) :
  `Public Sub NomMacro(arguments)`
- La procédure privée commence par *Private* :
  `Private Sub NomMacro(arguments)`

De plus, il est possible de définir des arguments afin d'utiliser une valeur d'une variable de procédure vers une sous-procédure.

PROCÉDURE AVEC UNE SOUS-PROCÉDURE

Procédure *ChoixPays* qui vérifie si la cellule active contient bien du texte. Si c'est le cas, elle lance la sous-procédure *AffichagePays* avec la cellule active comme valeur obligatoire et la cellule à droite de la cellule active comme valeur facultative (Optional)

Sous-procédure

```
Private Sub AffichagePays(Pays As String, Optional Capital As String)
    ' Sous-procédure qui affiche le pays et, éventuellement, sa capitale
    MsgBox "Pays choisi : " & Pays & Chr(10) & Capital
End Sub
```

Procédure principale (à exécuter pour lancer également la sous-procédure)

```
Sub ChoixPays()
    If VarType(ActiveCell) = vbString Then
        AffichagePays ActiveCell, ActiveCell.Offset(0, 1)
    End If
End Sub
```

TYPE D'APPEL DE LA VARIABLE

La sous-procédure peut modifier ou non la variable d'origine :

- **ByRef** : fait des calculs qui changent la variable d'origine (valeur par défaut)
- **ByVal** : fait des calculs qui ne modifient pas la variable d'origine

La procédure et la sous-procédures qui suivent augmentent ou diminuent la valeur de base de 10% mais avec *ByVal*, la valeur de la procédure d'origine sera modifiée ce qui ne sera pas le cas avec *ByRef* :

Sous-procédures et procédure principale

Sous-procédure qui ne modifie pas la variable d'origine :

```
Private Sub Diminution(ByVal Montant As Single)
    ' Diminue la valeur de 10% mais ne change pas la variable Montant d'origine
    Montant = Montant * 0.9
End Sub
```

Sous-procédure qui modifie la variable d'origine :

```
Private Sub Augmentation(ByRef Montant As Single)
    ' Augmente la valeur de 10% mais modifie la variable Montant d'origine
    Montant = Montant * 1.1
End Sub
```

Procédure principale à exécuter :

```
Sub Progression()' Procédure principale
    Dim Ventes As Single
    Ventes = 10000
    Diminution Ventes ' la valeur Ventes restera à 1000 (ByVal)
    MsgBox "la valeur Ventes reste à 1000 et non 9000 (ByVal) : " & Ventes
    Augmentation Ventes ' la valeur Ventes passera à 11000 (ByRef)
    MsgBox "la valeur Ventes passe à 11000 (ByRef) : " & Ventes
End Sub
```

✓ Lors de l'appel de la sous-procédure *Diminution*, le contenu de la variable affichée dans la procédure principale à la fin sera le même qu'au départ car le contenu de la variable *Ventes* n'est pas modifié. Si c'est la sous-procédure *Augmentation* qui est appelée, le contenu de la variable *Ventes* sera augmenté de 10% (à travers la variable *Montant*).

APPEL D'UNE PROCÉDURE D'UN MODULE À UN AUTRE

Dans le même module (ou si ce nom est unique) :

Il suffit de mettre le nom de la procédure, exemple :

```
Export ' Lance la procédure export située dans le même module
```

D'un module à un autre module :

Si le nom de la procédure existe dans plusieurs modules, il faut préciser le nom du module :

`Suivi.MAJ` ou `Call Suivi.MAJ ' Nom du module.Nom de la procédure`

La procédure *MAJ* est appelée dans le module ayant pour nom *Suivi*.

Pour appeler une macro d'un classeur vers un autre classeur :

Écrire Application.Run suivi du classeur, du module et de la procédure :

```
Application.Run "'Tableau des contacts.xlsm'!Formulaire.Ouverture"
```

✓ Ouvre la procédure *Ouverture*, dans le module *Formulaire* située dans le classeur *Tableau des contacts.xlsm*

LIENS SUR LES PROCÉDURES ET SOUS-PROCÉDURES

Création de procédures publiques et privées :
http://www.excel-pratique.com/fr/vba/procedures_fonctions.php

Lien vers les procédures publiques et privées ainsi que les variables :
https://openclassrooms.com/courses/analysez-des-donnees-avec-excel/modules-fonctions-et-sous-routines