Utiliser les variables en VBA Excel

par SilkyRoad (silkyroad.developpez.com)

Date de publication : 14/09/2006

Dernière mise à jour : 09/05/2007

Ce tutoriel présente l'utilisation des variables en VBA Excel. Une partie des informations est issue de l'aide en ligne Excel.

I - Introduction

Les variables servent à stocker et manipuler des informations dans une macro.

Une variable possède:

* Un nom qui permet d'accéder aux données qu'elle contient: "maVariable".

* Un type de données: Par exemple, String et Integer dans les exemples ci dessous.

La macro ci dessous affecte une chaîne de caractères (Bonjour !) dans une variable (maVariable), puis l'affiche dans un MsgBox:

Vba
```
Option Explicit

Sub afficherMessage()
    Dim maVariable As String

    maVariable = "Bonjour !"
    MsgBox maVariable
End Sub
```

Les données peuvent être modifiées pendant l'éxécution de la macro.

Un exemple pour affecter une valeur numérique à une variable, l'afficher dans un MsgBox, ajouter la valeur de la cellule A1 à cette variable, puis afficher le nouveau résultat:

Vba
```
Sub afficherValeur()
    Dim maVariable As Integer

    maVariable = 10
    MsgBox maVariable

    maVariable = maVariable + Range("A1")
    MsgBox maVariable
End Sub
```

Evitez de nommer les variables en utilisant des mots clés ou instructions réservés par Excel (par exemple Val, Left...).

Le nom des variables doit commencer par un caractère alphabétique et pas excéder 255 caractères. Les noms ne doivent pas contenir de caractères spéciaux. Le caratère underscore _ est accepté. Essayez de donner des noms les plus explicites possibles afin de faciliter la relecture de votre programme.

Il est conseillé d'avoir au moins une majuscule dans la variable déclarée. Ensuite lors de la saisie de la variable en minuscule dans la macro, celle-ci reprendra automatiquement la majuscule: cette astuce permet de vérifier les fautes d'orthographe éventuelles.

Par exemple: Dim RepertoireFichier As String.

Attribuez des noms explicites qui précisent le type et l'utilisation de la variable, afin de faciliter la relecture de votre code.

Par exemple: Dim StrCouleur As String

Str sert à indiquer qu'il s'agit d'un type **String**.

Le type de données doit être défini en fonction de la valeur prise par la variable. Chaque type de donnée utilise un espace mémoire (de 1 octet pour les types de données Byte jusqu'à 22 octets et plus, pour les types de données Variant). Il est donc important de définir le bon type de données pour libérer de l'espace mémoire et ne pas ralentir inutilement le traitement de la macro.

Le chapitre suivant décrit les types de données disponibles.

II - Les types de données

Les informations de ce chapitre sont en grande partie issues de l'aide en ligne Excel.

Toutes les variables sont converties en type Variant si aucun autre type de données n'est explicitement déclaré.

En cas de déclaration de plusieurs variables avec le même Dim, vous devez préciser le type de donnée pour chaque variable.

Par exemple, si pour définir 3 variables des type String (strVar1, strVar2 et strVar3) vous écrivez:

Dim strVar1 , strVar2 , strVar3 As String

Dans ce cas strVar1 et strVar2 seront de type Variant.

Pour y remédier et obtenir 3 variables String, Il faut écrire:

Dim strVar1 As String , strVar2 As String , strVar3 As String

II-A - Byte

Les variables Byte sont utilisées pour stocker des nombres entiers positifs compris entre 0 et 255.

Les variables de type Byte sont stockées sous la forme de nombres uniques codés sur 8 bits (1 octet), sans signe.

II-B - Boolean

Données pouvant prendre exclusivement les valeurs True et False.

Les variables Boolean sont stockées sous la forme de nombres codés sur 16 bits (2 octets).

II-C - Integer

Données contenant des nombres entiers stockés, de 2 octets, compris entre -32 768 et 32 767.

Le type de données Integer permet également de représenter des valeurs énumérées.

Dans Visual Basic, le signe % est le caractère de déclaration du type Integer.

Remarque:

Si vous écrivez "Dim X As Integer", alors que la valeur est décimale (par exemple X=5,9), la valeur renvoyée sera égale à 6.

II-D - Long

Nombre entier codé sur 4 octets (32 bits) et dont la valeur est comprise entre -2 147 483 648 et 2 147 483 647.

Dans Visual Basic, le signe et commercial (&) est le caractère de déclaration du type Long.

II-E - Currency

Données dont la plage de valeurs s'étend de -922 337 203 685 477,5808 à 922 337 203 685 477,5807.

Les variables de type Currency sont stockées sous la forme de nombres de 64 bits (8 octets).

Ce type de données est utilisé dans les calculs monétaires ou dans les calculs à virgule fixe pour lesquels une grande précision est requise. Le signe @ est le caractère de déclaration du type Currency.

II-F - Single

Type de données qui regroupe des variables à virgule flottante en simple précision sous forme de nombres à virgule flottante codés sur 32 bits (4 octets), dont la valeur est comprise entre -3,402823E38 et -1,401298E-45 pour les valeurs négatives, et entre 1,401298E-45 et 3,402823E38 pour les valeurs positives.

Dans Visual Basic, le point d'exclamation (!) est le caractère de déclaration du type Single.

II-G - Double

Type de données stockant sur 64 bits les nombres à virgule flottante en double précision compris entre -1,79769313486231E308 et -4,94065645841247E-324 pour les valeurs négatives, et entre 4,94065645841247E-324 et 1,79769313486232E308 pour les valeurs positives.

Dans Visual Basic, le signe dièse (#) est le caractère de déclaration du type Double.

II-H - Date

Type de données utilisé pour stocker les dates et les heures sous la forme d'un nombre réel codé sur 64 bits (8 octets). La partie située à gauche du séparateur décimal représente la date, et la partie droite l'heure.

II-I - String

Type de données composé d'une séquence de caractères contigus interprétés en tant que caractères et non en tant que valeurs numériques.

Une donnée de type String peut inclure lettres, nombres, espaces et signes de ponctuation.

Le type de données String peut stocker des chaînes de longueur fixe dont la longueur est comprise entre 0 et environ 63 Ko de caractères et des chaînes dynamiques dont la longueur est comprise entre 0 et environ 2 milliards de caractères.

Dans Visual Basic, le signe dollar ($) est le caractère de déclaration du type String.

II-J - Variant

Type de données particulier pouvant contenir des données numériques, des chaînes ou des dates, des types définis par l'utilisateur ainsi que les valeurs spéciales Empty et Null.

Le type de données Variant est doté d'une taille de stockage numérique de 16 octets et peut contenir la même plage de données que le type Decimal, ou d'une taille de stockage de caractère de 22 octets (plus la longueur de la chaîne). Dans ce dernier cas, il peut stocker tout texte.

II-K - Object

Type de données représentant toute référence Objet. Les variables Object sont stockées sous forme d'adresses codées sur 32 bits (4 octets). L'instruction Set permet d'attribuer une référence d'objet à la variable.

Dans Excel, un objet peut être un classeur, les feuilles de calcul, un graphique...etc...

La macro suivante déclare la variable Ws comme un objet de type Worksheet (Feuille de cacul).

Vba

```
Sub Test_V01()
    Dim Ws As Worksheet

    'Attribue la 1ere feuille du classeur dans la variable
    Set Ws = Sheets(1)
    MsgBox Ws.Name

    Set Ws = Nothing 'Libère la mémoire
End Sub
```

Vous pouvez aussi utiliser la syntaxe ci-dessous.

Nota: Vous perdez l'avantage de la saisie semi automatique en utilisant cette méthode.

Vba

```
Sub Test_V02()
    Dim Ws As Object

    'Attribue la 1ere feuille du classeur dans la variable
    Set Ws = Sheets(1)
    MsgBox Ws.Name

    Set Ws = Nothing 'Libère la mémoire
End Sub
```

Vous avez aussi la possiblité de communiquer avec d'autres applications depuis Excel: **Word**, **ADO (ActiveX Data Objects)**,...

Vous pouvez dans ce cas activer les références afin de manipuler les objets.

Voici un exemple de syntaxe pour utiliser les objets Word depuis Excel.

Vba

```
'Vous devez préalablement activer la référence Word:
  'Dans l'éditeur de macros:
  'Menu outils
  'Références
  'Cochez la ligne "Microsoft Word x.x Object Library"
  '(x.x dépend de la version installée sur votre poste)
  'Cliquez sur OK pour valider.

Sub piloterWord_V01()
    Dim wordApp As Word.Application 'déclare la variable wordApp comme un objet de type Word
    'Remarque:
    'Lorsque la librairie est activée,l'outil de saisie semi automatique permet
    'd'afficher la bibliotheque Word sans avoir besoin de saisir le nom complet.
    'Toutes les méthodes et propriétés de la librairie sont aussi accessibles grace
    'à l'outil de saisie semi automatique.

    'Attribue la référence objet à la variable
    Set wordApp = New Word.Application

    MsgBox "La version Word installée sur votre poste: " & wordApp.Version

    'Fermeture de l'application Word
    wordApp.Quit
    'libération de la mémoire
    Set wordApp = Nothing
End Sub
```

Il est important de libérer l'espace mémoire en fin de procédure: Set wordApp = Nothing

La configuration WindowsXP/Office 97 peut provoquer des erreurs lors de la déclaration des objets.

Le message qui s'affiche est Erreur d'éxecution -2147417851 (80010105)

Une solution consiste à utiliser une liaison tardive et de mofiier la déclaration de variables ainsi:

Vba

```
Sub piloterWord_V02()
    'Plus d'informations sur le site Microsoft
    'http://support.microsoft.com/?id=242375
    Dim wordApp As Object

    'Attribue la référence objet à la variable
    Set wordApp = CreateObject("Word.Application")

    MsgBox "La version Word installée sur votre poste: " & wordApp.Version

    'Fermeture de l'application Word
    wordApp.Quit
```

```Vba
        'libération de la mémoire
        Set wordApp = Nothing
    End Sub
```

Remarque:

Vous n'aurez pas la possibilité d'utiliser les constantes des librairies à partir des liaisons tardives. Vous devrez les remplacer par leur valeur.

III - Option Explicit

L'instruction Option Explicit est utilisée au niveau module pour imposer la déclaration explicite de toutes les variables de ce module. Cette instruction doit apparaître tout en haut dans le module, avant toute procédure.

Lorsque cette instruction est utilisée, un message d'erreur identifie toute variable non définie ou mal orthographiée.

Pour qu'Option Explicit s'insère automatiquement dans chaque nouveau classeur:

Allez dans l'éditeur de macros.

Menu Outils

Options

Dans l'onglet Editeur, cochez l'option "Déclaration Explicite des variables".

IV - La différence entre ByRef et ByVal

IV-A - ByRef

ByRef permet de passer à une procédure l'adresse d'un argument plutôt que sa valeur. La procédure peut ainsi accéder à la variable proprement dite. La valeur réelle de cette dernière peut, de ce fait, être modifiée par la procédure à laquelle elle a été passée. Par défaut, les arguments sont passés par référence.

Si la procédure appelée change la valeur de ces variables, elle changeront au retour dans la procédure appelante.

IV-B - ByVal

ByVal permet de passer à une procédure la valeur d'un argument plutôt que son adresse. La procédure peut de ce fait accéder à une copie de la variable. La valeur réelle de cette dernière n'est donc pas modifiée par la procédure à laquelle elle est passée.

Si la procédure appelée change la valeur des variables, elles ne changeront pas dans la procédure appelante.

L'utilisation de ByVal implique un temps de calcul plus long et nécessite un espace mémoire plus important.

IV-C - Exemple

Un exemple de mise en application pour montrer les valeurs prises successivement par la variable **Donnee**, en fonction du passage par des sous procédures **ByRef**, **ByVal**, puis non spécifiée.

Vba

```
Sub Test()
    Dim Donnee As Integer

    Donnee = 50

    MaProcedure_1 Donnee
    MsgBox Donnee

    MaProcedure_2 Donnee
    MsgBox Donnee

    MaProcedure_3 Donnee
    MsgBox Donnee
End Sub


'Passe la référence en argument.
Sub MaProcedure_1(ByRef x As Integer)
    x = x * 2
End Sub


'Passe la valeur en argument.
Sub MaProcedure_2(ByVal y As Integer)
    y = y * 2
End Sub
```

Vba
```
'ByRef est la valeur par défaut si non spécifiée.
Sub MaProcedure_3(z As Integer)
    z = z * 2
End Sub
```

Cas particulier **ByRef**, si la variable est encadrée par des parenthèses:

Dans ce cas, si la procédure appelée change la valeur de la variable, elle ne changera pas dans la procédure appelante.

Vba
```
Sub Test()
    Dim i As Integer

    i = 1

    Essai (i)
    MsgBox i

    Essai i
    MsgBox i
End Sub


Sub Essai(ByRef j As Integer)
    j = j + 1
End Sub
```

V - Les niveaux de variables

Une Variable déclarée à l'intérieur d'une macro ne sera utilisable qu'à l'intérieur de celle-ci.

Vba
```
Sub Test()
    Dim X As String

    X = "Coucou ! "
    MsgBox X
End Sub
```

Pour que la variable soit utilisable dans toutes macros du module, celle-ci doit être déclaréee en tête du module, avant le premier Sub.

Vba
```
Dim X As String

Sub Test()
#
End Sub
```

Pour que la variable soit utilisable dans toutes les macros du projet, il faut utiliser l'instruction Public et la variable doit impérativement être placée en tête d'un module standard.

Vba
```
Public X As String

Sub Test()
...
End Sub
```

VI - Les instructions Const et Enum

VI-A - Const

L'instruction Const permet de déclarer les constantes.

Une constante est un élément nommé conservant une valeur identique pendant toute l'exécution d'un programme. Il peut s'agir d'une chaîne, d'une donnée numérique, d'une autre constante ou d'une combinaison contenant des opérateurs logiques ou arithmétiques à l'exception de Is et de l'opérateur d'élévation à une puissance. Les constantes peuvent remplacer des valeurs réelles partout dans votre code.

Il y a plusieurs avantages à utiliser une constante:

Si la donnée doit être modifiée dans une macro complexe, vous n'avez plus besoin de parcourir toute la procédure pour la retrouver: il suffit de modifier la constante qui est généralement placée en début de macro ou en tête du module. Vous évitez ainsi les recherches fastidieuses et les erreurs de saisie.

Les constantes permettent aussi d'améliorer la lisibilité des macros.

Un exemple d'utilisation:

Vba
```
Sub Test()
    Const Coefficient As Integer = 10
    Dim x As Integer

    x = Coefficient * 5
    MsgBox x
End Sub
```

VI-B - Enum

L'instruction ENUM permet de définir un groupe de constantes liées afin de créer une énumération.

Un exemple:

Vba
```
Public Enum Coeff
    Coeff_2 = 2
    Coeff_3 = 3
    Coeff_4 = 4
End Enum


Sub Test()
    MsgBox 500 * Coeff.Coeff_2 'Résultat = 1000
```

```vba
    End Sub
```

Remarque:

La saisie de Coeff. dans l'éditeur de macros, permet d'afficher rapidement la liste des coefficients disponibles.

VII - Les fonctions de conversion des types de données

Ces fonctions convertissent une expression en un type de données spécifié.

Elles sont particulièrement intéressantes pour réadapter un format, par exemple suite à un transfert de données brutes que vous souhaitez utiliser en type Date, numérique... De la même manière, les données saisies dans des **TextBox** ont souvent besoin d'être converties pour une utilisation dans la suite du programme.

Fonction	Type renvoyé	Plage de valeurs
CBool	Boolean	Toute chaîne ou expression numérique valide
CByte	Byte	0 à 255
CCur	Currency	922 337 203 685 477,5808 à 922 337 203 685 477,5807
CDate	Date	Toute expression de date valide
CDbl	Double	-1.79769313486231E308 à -4,94065645841247E-324 pour les valeurs négatives ; 4,94065645841247E-324 à 1,79769313486232E308 pour les valeurs positives
CDec	Decimal	+/-79 228 162 514 264 337 593 543 950 335 pour les nombres sans décimales. La plage de valeurs des nombres à 28 décimales est +/-7,9228162514264337593543950335. Le plus petit nombre différent de zéro est 0,0000000000000000000000000001
CInt	Integer	-32 768 à 32 767 ; les fractions sont arrondies. Les parties décimales sont arrondies à 0,5 au nombre pair le plus proche
CLng	Long	2 147 483 648 à 2 147 483 647 ; les fractions sont arrondies. Les parties décimales sont arrondies à 0,5 au nombre pair le plus proche
CSng	Single	-3,402823E38 à -1,401298E-45 pour les valeurs négatives ; 1,401298E-45 à 3,402823E38 pour les valeurs positives
CStr	String	Les valeurs renvoyées par la fonction Cstr dépendent de l'argument expression
CVar	Variant	Même plage de valeurs que le type Double pour les nombres et que le type String pour les chaînes non numériques

Un exemple qui transforme une chaîne de caractères de type String en type Date.

Vba

```
Sub Essai()
    Dim maVariable As String

    maVariable = "26/05/2005"

    'Ecrit le contenu de la variable dans la cellule A1
    Range("A1") = maVariable
    'Ecrit la variable convertie en date dans la cellule A2
    Range("A2") = CDate(maVariable)
End Sub
```

VIII - Les fonctions TypeName, VarType et la clause TypeOf

VIII-A - TypeName

La fonction TypeName permet de récupérer des informations sur une variable.

Vba

```
'Boucler sur les contrôles d'un UserForm et vérifier s'il s'agit de TextBox
Dim Ctrl As Control

For Each Ctrl In Userform1.Controls
    If TypeName(Ctrl) = "TextBox" Then
    '
    '...
    '
    End If
Next Ctrl
```

Vba

```
'Vérifier si la variable Nbr est de type Integer
If TypeName(Nbr) = "integer" Then
```

Vba

```
'Vérifier le type de donnée contenu dans la cellule A1
MsgBox TypeName(Range("A1").Value)
```

VIII-B - VarType

La fonction VarType permet de renvoyer le sous-type d'une variable type variant.

Constante	Valeur	Description
vbEmpty	0	Empty (non initialisée)
vbNull	1	Null (aucune donnée valide)
vbInteger	2	Entier
vbLong	3	Entier long
vbSingle	4	Nombre à virgule flottante en simple précision
vbDouble	5	Nombre à virgule flottante en double précision
vbCurrency	6	Valeur monétaire
vbDate	7	Valeur de date
vbString	8	Chaîne
vbObject	9	Objet
vbError	10	Valeur d'erreur
vbBoolean	11	Valeur booléenne
vbVariant	12	Variant (utilisée seulement avec des tableaux de variants)
vbDataObject	13	Objet d'accès aux données

vbDecimal	14	Valeurs décimales
vbByte	17	Octet
vbUserDefinedType	36	Variant contenant des types définis par l'utilisateur
vbArray	8192	Tableau

Quelques exemples d'utilisation:

Vba
```
'Intercepter l'utilisation du bouton "Annuler" et la croix de fermeture d'un Inputbox
    Dim Reponse As Variant
    Reponse = Application.InputBox("Saisissez vos données")
    If VarType(Reponse) = vbBoolean Then MsgBox " opération annulée"
```

Vba
```
Sub Test()
    Dim maVariable As Variant

    maVariable = "Coucou"
    MsgBox VarType(maVariable) 'Renvoie 8

    maVariable = 123
    MsgBox VarType(maVariable) 'Renvoie 2

    maVariable = #5/26/2005#
    MsgBox VarType(maVariable) 'Renvoie 7
End Sub
```

Vba
```
Sub TestTableau()
    Dim Tableau(3)

    Tableau(0) = 1234
    Tableau(1) = "mimi"

    MsgBox VarType(Tableau(0)) 'renvoie 2
    MsgBox VarType(Tableau(1)) 'renvoie 8
    MsgBox VarType(Tableau(2)) 'renvoie 0
End Sub
```

VIII-C - TypeOf

La clause TypeOf permet de contrôler vers quel objet de l'application pointe la variable. L'objet peut être une feuille, une cellule, un contrôle...

Quelques exemples d'utilisation:

Vba
```
'Retrouver le nom de toutes les feuilles graphiques dans le classeur actif.
 Dim Sh As Object
 For Each Sh In ActiveWorkbook.Sheets
 If TypeOf Sh Is Chart Then MsgBox Sh.Name
```

Vba
```
        Next
```

Vba
```
    'Vérifier si une cellule est sélectionnée dans la feuille
     If typeOf Selection Is Range Then
```

Vba
```
    'Boucler sur les CheckBox d'un UserForm et leur attribuer la valeur True
     Dim Ctrl As Control
     For Each Ctrl In Me.Controls
     If TypeOf Ctrl Is MSForms.checkBox Then Ctrl.Value = True
     Next
```

IX - Les types de données définis par l'utilisateur

Vous pouvez utiliser l'instruction Type afin de déclarer un type de données personnalisé.

Les types définis par l'utilisateur peuvent contenir un ou plusieurs éléments de n'importe quel type de données. Il est possible de créer des types adaptés à un projet pour en faciliter le traitement.

Les types placés dans un module standard sont Public par défaut. Ils peuvent être redéfinis en Private. Les types utilisés dans les modules de classe sont toujours Private et ne peuvent pas être modifiés en Public.

Un exemple:

Vba

```
'Ces lignes sont sont à placer en tête du module
Type Voiture
    Couleur As String
    Cylindree As Long
    anneeAchat As Date
End Type


Sub Test()
    Dim X As Voiture 'Déclaration de la variable personnalisée

    X.Couleur = "Rouge" 'Attribution des données
    X.Cylindree = 2000
    X.anneeAchat = #4/21/2004#

    MsgBox "Cette voiture " & X.Couleur & " a une cylindrée de " & _
        X.Cylindree & " cc, Année " & X.anneeAchat 'lecture des données
End Sub
```

Il est aussi possible d'utiliser un tableau pour stocker les données.

Vba

```
Sub Test_V02()
    Dim Tableau(20) As Voiture

    'Remplissage de 1ere ligne Tableau
    Tableau(0).Couleur = "Rouge"
    Tableau(0).Cylindree = 2000
    Tableau(0).anneeAchat = #4/21/2004#

    'Lecture contenu tableau
    MsgBox "Cette voiture " & Tableau(0).Couleur & " a une cylindrée de " & _
        Tableau(0).Cylindree & " cc, année " & Tableau(0).anneeAchat
End Sub
```

X - Téléchargement