VBA et les variables
Catégorie : VBA pour débutants
Publié par myDearFriend! le 09-02-2008

Déclarer les variables dans VBA ?

La déclaration des variables est toujours recommandée.

Il s'agit de la toute première étape en terme d'optimisation du code VBA.

Bien que non obligatoire, la déclaration des variables avec le type de données adéquat apporte des avantages indéniables :

1. Plus vous rationaliserez les types de variable utilisées et mieux vous maitriserez le déroulement de votre projet VBA (savoir ce qu'on fait et comment on y parvient, est déjà le premier pas vers la maitrise, non ?).
2. L'objectif est avant tout d'optimiser les ressources mémoire utilisées et d'accroitre ainsi les performance de votre projet.
3. Le plus souvent, le choix du bon type de variable accentue la vitesse d'exécution de façon perceptible.
4. L'utilisation de l'instruction Option Explicit en tête de module facilite grandement le débogage de votre projet.

Les Types de Données VBA

Types de données	Plages de valeurs permises	Poids
Byte	de 0 à 255	1 octet
Boolean	**True** ou **False**	2 octets
Integer	de -32 768 à 32 767	2 octets
Long (entier long)	de -2 147 483 648 à 2 147 483 647	4 octets
Single (à virgule flottante en simple précision)	de -3,402823E38 à -1,401298E-45 pour les valeurs négatives de 1,401298E-45 à 3,402823E38 pour les valeurs positives	4 octets
Double (à virgule flottante en double précision)	de -1,79769313486231E308 à -4,94065645841247E-324 pour les valeurs négatives de 4,94065645841247E-324 à 1,79769313486232E308 pour les valeurs positives	8 octets
Currency (entier à décalage)	de -922 337 203 685 477,5808 à 922 337 203 685 477,5807	8 octets
Decimal	+/- 79 228 162 514 264 337 593 543 950 335 sans séparateur décimal ; +/- 7,9228162514264337593543950335 avec 28 chiffres à droite du séparateur décimal ; le plus petit nombre différent de zéro est +/- 0.0000000000000000000000000001.	14 octets
Date	1er janvier 100 au 31 décembre 9999	8 octets
Object	Toute référence à des données de type **Object**	4 octets
String (longueur variable)	0 à environ 2 milliards	10 octets + longueur de la chaîne
String (longueur fixe)	1 à environ 65 400	Longueur de la chaîne
Variant (nombres)	Toute valeur numérique, avec la même plage de valeurs qu'une donnée de type **Double**	16 octets
Variant (caractères)	Même plage de valeurs qu'une donnée de type **String** de longueur variable	22 octets + longueur de la chaîne

Pour déclarer correctement une variable, l'objectif est simple : choisir le type de données le plus approprié aux valeurs à stocker dans cette variable tout en privilégiant le type représentant le poids le plus faible.

Quelques exemples illustrant ce principe :

- Pour stocker un simple état VRAI ou FAUX d'un élément quelconque, je choisirais le type de données Boolean (valeurs True / False).

```
Dim EtatOK As Boolean
```

- Pour stocker le numéro du jour (1-31) dans une variable, je choisirais le type de données Byte (valeur pouvant aller de 0 à 255).

```
Dim Jour As Byte
```

- Pour stocker le numéro de ligne de la sélection courante d'une feuille de calcul, je choisirais le type Long (une feuille de calcul comporte 65536 lignes - et même 1 048 576 depuis la dernière version Excel 2007).

```
Dim Lign As Long
```

EXCEPTIONS A LA REGLE !

- Type Integer ou type Long ?
  Traditionnellement, les développeurs VBA utilisent le type Integer lorsque les valeurs à stocker le permettent, parce que ce type de données requiert moins de ressources mémoire que le type Long (c'est même la règle du jeu expliquée ci-dessus).
  Toutefois, une information d'importance semble méconnue : aujourd'hui VBA convertit automatiquement en interne tous les éléments de type Integer en élément de type Long, et ce, même si la déclaration explicite est "As Integer". Il est donc devenu préférable aujourd'hui d'utiliser le type Long en lieu et place du type Integer pour accroître les perfomances du code. En fait, le traitement des variables de type Long est même devenu plus rapide car ce type de données n'a pas besoin d'être converti avant traitement !

- Type Currency ou type Single ?
  Tout d'abord, il convient d'éviter les types de variable à virgule flottante. Par ailleurs, bien que plus gourmand en ressources (poids plus important), le type Currency est à privilégier au type Single quand c'est possible. En effet, le type Currency n'utilise pas le système de calcul à virgule flottante et le traitement s'en trouve accéléré.

- Type Object ?
  En terme de gain de performance, il est toujours préférable d'assigner un objet au type de variable spécifique correspondant, plutôt qu'au type générique Object.
  Par exemple, il vaut mieux faire "Dim MaFeuille As Worksheet" plutôt que "Dim MaFeuille As Object".

Le Nom des variables

Vous pouvez utiliser n'importe quel expression pour nommer vos variables. Pour des raisons évidentes, il convient toutefois de choisir une expression explicite qui vous aidera à mieux appréhender votre code lors de modifications ou débogages futurs.

Notez cependant les restrictions suivantes (selon les règles d'affectation de noms applicables aux variables, mais aussi aux procédures, constantes et arguments dans VBA) :

- Utilisez une lettre comme premier caractère.
- N'utilisez pas d'espace, de point (.), de point d'exclamation ! ou les caractères @, &, $, #

dans le nom.
- Un nom ne peut compter plus de 255 caractères.
- Vous ne devez pas utiliser des noms identiques aux noms de fonction, d'instruction et de méthode de Visual Basic. Vous feriez double emploi des mots clés du langage.
- Vous ne pouvez pas employer deux fois le même nom au même niveau de portée. Par exemple, vous ne pouvez pas déclarer deux variables nommées MaVar dans la même procédure. Cependant, vous pouvez déclarer une variable privée nommée MaVar et une variable de niveau procédure nommée MaVar dans le même module.

Remarque : VBA ne différencie pas les majuscules des minuscules, mais conserve la casse issue de l'instruction de déclaration.

Option Explicit

L'utilisation de l'instruction Option Explicit placée en tête de module impose la déclaration explicite (par Dim, Private, Public, ReDim ou Static) de toutes les variables de ce module. Cette instruction doit apparaître avant toute procédure dans le module.
Avec cette instruction, si vous tentez d'utiliser un nom de variable non déclarée, une erreur se produit à la compilation. Utilisez donc Option Explicit pour éviter toute faute de frappe lors de la saisie du nom d'une variable existante et pour écarter tout risque de confusion dans un code où la portée de la variable n'apparaît pas clairement.

Si vous n'utilisez pas cette instruction Option Explicit, toutes les variables non déclarées seront considérées comme Variant (c'est à dire le type de données au poids le plus lourd !).

Utiliser Option Explicit par défaut

 Et pour ne pas avoir à mettre manuellement Option Explicit dans chaque nouveau module et ou projet que vous insérez, vous pouvez, depuis l'éditeur VBE, faire Menu Outils / Options.../ Onglets "Editeur" et cocher "Déclaration des variables obligatoire".

Portée et visibilité des variables

La portée se réfère à la disponibilité d'une variable pour son utilisation dans le code. On choisit la portée d'une variable lors de sa déclaration.

Il existe trois niveaux de portée : niveau de Procédure, niveau de Module privé et niveau de Module public.

Les variables de niveau module utilisant des ressources mémoire jusqu'à la réinitialisation de leurs valeurs. Pour optimiser au mieux votre code et économiser les ressources machine, il convient de les utiliser uniquement lorsque cela est nécessaire.

1) Portée de niveau PROCEDURE
C'est la façon la plus courante de déclarer une variable. On utilise ici l'instruction de déclaration Dim. Une variable définie dans une procédure n'est pas visible à l'extérieur de celle-ci. Seule la procédure contenant la déclaration de variable peut l'utiliser. La déclaration s'effectue juste après l'instruction Sub...().

Exemple :

DANS UN MODULE DE CODE

```
Option Explicit

Sub MaProc()
    Dim MaVar As String     'Variable déclarée en tête d'une procédure
    MaVar = "Cette variable ne peut pas être utilisée en dehors de cette procédure."
    MsgBox MaVar
End Sub
```

2) Portée de niveau MODULE PRIVE

Les variables de cette portée sont déclarées en tête de module (avant toute procédure, mais après l'instruction Option Explicit si celle-ci est utilisée). On utilisera ici les instructions de déclaration Dim ou Private (dans ce cas de figure, Dim et Private ont la même signification et vous pouvez utiliser Private pour améliorer la lisibilité de votre code).

Une variable de niveau module, déclarée à l'aide de l'instruction Dim (ou Private), est utilisable et visible depuis l'ensemble des procédures contenues dans ce module. Elle ne sera pas utilisable dans d'autres modules du projet.

Exemple :

DANS UN MODULE DE CODE

```
Option Explicit

Dim MaVar As String     'Variable déclarée en tête d'un module

Sub MaProc1()
    MaVar = "Cette variable peut être utilisée dans tout le module."
End Sub

Sub MaProc2()
    MsgBox MaVar
End Sub
```

3) Portée de niveau MODULE PUBLIC

Le principe reste le même que pour la portée de niveau Module Privé. La différence est que nous utiliserons l'instruction de déclaration Public (à la place de Dim ou Private) dans ce cas.

Une variable de niveau module, déclarée à l'aide de l'instruction Public, est utilisable et visible dans le projet tout entier. On pourra appeler cette variable depuis n'importe quelle procédure de n'importe quel module du projet en question.

> Particularité :
> Si vous faites cette déclaration public en tête d'un module de code standard (ex: Module1), il n'y a aucune particularité pour appeller cette variable ailleurs dans le code, il suffit simplement d'y faire référence par son nom. Par exemple : MsgBox MaVar
>
> Par contre, si vous déclarez public une variable depuis l'en-tête d'un module de code évènementiel (module de code d'une feuille ou de l'objet ThisWorkbook par exemple), l'appel de cette variable devra explicitement faire référence au module de code évènementiel en question. Ainsi, pour appeler une variable qui a été déclarée public dans l'entête du module de code de la feuille 1 par exemple, il conviendra de faire : Msgbox Feuil1.MaVar

Durée de vie des variables

La période pendant laquelle une variable conserve sa valeur correspond à sa durée de vie. La valeur d'une variable peut changer pendant sa durée de vie, mais elle conserve une certaine valeur. Lorsqu'une variable perd sa portée, elle n'a plus de valeur.

Au début d'une procédure, toutes les variables sont initialisées dès leur déclaration : une variable numérique est initialisée à zéro, une chaîne de longueur variable est initialisée à une chaîne de longueur nulle (""), et une chaîne de longueur fixe est remplie du caractère représentant le code de caractères ASCII 0, ou Chr(0). Les variables de type Variant prennent initialement la valeur Empty. Lorsque vous déclarez une variable objet, de l'espace est réservé en mémoire, mais elle prend la valeur Nothing jusqu'à ce que vous lui affectiez une référence d'objet à l'aide d'une instruction Set.

Si la valeur d'une variable n'est pas modifiée pendant l'exécution du code, elle conserve sa valeur initialisée jusqu'à ce qu'elle perde sa portée.

Une variable de niveau procédure déclarée avec l'instruction Dim conserve une valeur jusqu'à la fin de la procédure.

Cas particulier des variables déclarées Static

A l'instar de Dim, l'instruction Static est utilisée au niveau Procédure pour déclarer des variables dont la portée s'arrête à la dite procédure.
Cependant, la durée de vie diffère de Dim : les variables déclarées à l'aide de l'instruction Static conservent leur valeur entre les différents appels de la procédure.

En résumé, une variable Static possède la portée d'une variable de niveau Procédure, mais sa durée de vie est comparable à une variable de niveau Module.

Exemple :

```
Sub Increment()
Static Compteur As Long
    Compteur = Compteur + 1
    MsgBox Compteur
End Sub
```

Lors de chaque appel de la procédure Increment() ci-dessus, la variable Compteur sera incrémentée d'une unité. La valeur de Compteur est donc mémorisée entre chaque appel de la procédure contrairement à une variable déclarée avec Dim qui est réinitialisée à chaque appel. Dans cet exemple, le comportement est similaire à l'utilisation d'une variable déclarée au niveau Module.

Alors, faut-il utiliser une variable Static ou une variable de portée Module dans ce cas particulier ?

Pour moi, la clé du problème se situe non seulement au niveau de la durée de vie de la variable mais aussi, et surtout, au niveau de la portée que l'on veut donner à la dite variable.

Une variable Public ou Private déclarée au niveau Module trouve son utilité uniquement si elle doit être partagée entre plusieurs procédures. Par exemple, si je décidais de chronométrer la durée d'ouverture d'un classeur, je déclarerais une variable vTemps en tête de module de l'objet ThisWorkbook, je lui affecterais l'heure courante dans l'évènement Open() et je réutiliserais sa valeur

dans l'évènement BeforeClose() du même objet pour calculer la durée de la session.

Dans le cas cité plus haut, il s'agit uniquement de permettre à la variable de conserver sa valeur. Une seule procédure est concernée ici et la valeur n'est pas partagée entre plusieurs évènements. La portée étant donc restreinte, une simple variable déclarée Static à l'intérieur même de cette procédure, suffit à répondre à la seule contrainte de durée de vie.

Personnellement, je privilégie l'utilisation des variables Static quand j'en ai la possibilité. Je les trouve beaucoup plus faciles à gérer. Pour moi, plus la portée de la variable est restreinte, et plus il devient facile d'en maîtriser le contenu. On réduit d'autant les risques de changement de valeur inattendu à un autre endroit dans le code...

On peut ainsi déclarer des variables Static avec le même nom dans plusieurs procédures différentes. Chacune de ses variables reste donc indépendante et conserve sa propre valeur dans sa propre procédure. Personnellement, je trouve ça pratique...