GESTION DES ERREURS

Excel VBA dispose de plusieurs fonctions permettant de mieux comprendre l'origine d'une erreur dans un code VBA. Celle-ci peut être détectée par un message d'alerte apparaissant lors de l'exécution de la macro mais il est également possible d'afficher une information sur le contenu de la variable, voire sur le type ou sur la description de l'erreur.

DÉBOGAGE D'UN PROGRAMME

La plupart des erreurs sont dues à une mauvaise valeur dans une variable (texte dans une variable de nombre, etc.) ou appeler un élément (variable, macro, fichier, procédure, etc.) qui n'existe pas. Excel dispose de plusieurs outils permettant de retrouver rapidement une erreur soit en arrêtant le programme sur la ligne erronée soit en proposant de surveiller une partie du programme, grâce notamment, aux fonctions de débogage de l'éditeur de Visual Basic.

RELANCER UNE MACRO

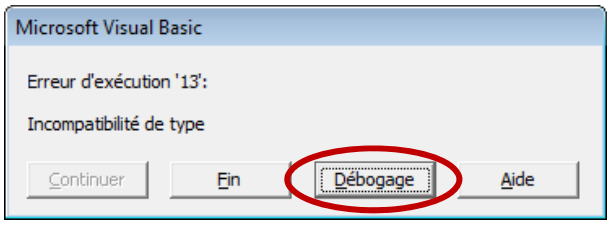
Lorsqu'une macro s'est arrêtée, il est nécessaire de la relancer :

Dans la fenêtre de Visual Basic, cliquer sur le bouton **Réinitialiser**. Le surlignage en jaune marquant l'arrêt de la macro disparaît.

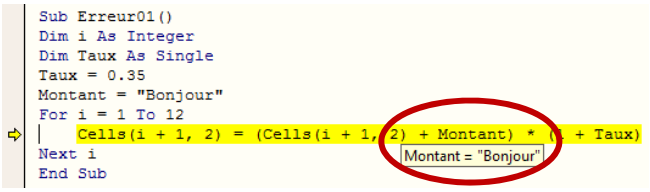
☞ À chaque arrêt de macro avant la fin du programme — suite à une erreur ou avec les instructions qui suivent par exemple — il est nécessaire de la réinitialiser, sinon celle-ci ne pourra plus être exécutée et certaines fonctions d'Excel ne seront plus accessibles.

RETROUVER UNE ERREUR D'INSTRUCTION

La plupart des erreurs sont dues à une mauvaise valeur dans une variable (texte dans une variable de nombre, etc.) ou appeler un élément (variable, macro, fichier, procédure, etc.) qui n'existe pas. Lorsqu'une instruction empêche le déroulement d'une macro, le programme est interrompu et un message d'erreur apparaît à l'écran :



1. Pour connaître la ligne ayant arrêté le code, cliquer sur le bouton **Débogage**
2. Le code apparaît, avec la ligne erronée est surlignée en jaune

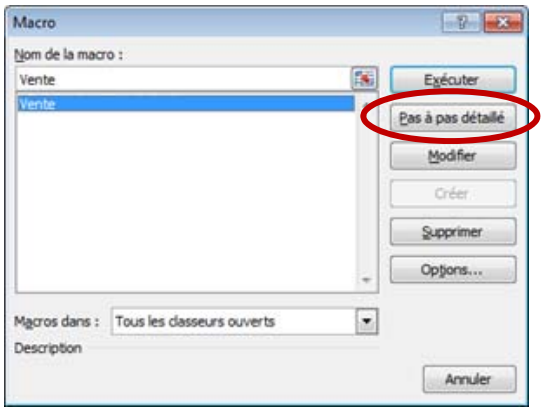


3. En pointant sans cliquer vers un nom de variable, son contenu apparaît dans une infobulle, ici *Montant= "Bonjour"* alors que cette variable devrait contenir un nombre
4. Dans son code, retrouver et corriger l'erreur en recherchant où est définie la valeur de la variable Montant : deux lignes au-dessus, remplacer *Montant="Bonjour"* par *Montant=100000*

DÉROULEMENT DU PROGRAMME PAS À PAS

La macro peut également se dérouler ligne par ligne, permettant de vérifier quelle ligne contient une erreur pour la corriger :

1. Onglet Développeur
   Macros
   Cliquer sur le nom de la macro voulue (exemple : Ventes)
   Cliquer sur **Pas à pas détaillé**
   ou, dans l'éditeur de Visual,Basic :
   Menu Débogage
   Pas à pas détaillé



2. Appuyer sur la touche de fonction **F8** pour exécuter les lignes de la macro une par une.
   La ligne en cours d'exécution apparaît, comme précédemment, surligné en jaune. Il est ainsi possible de vérifier la valeur prise par une variable.

```
Taux = 0.35
Montant = "Bonjour"
For i = 1 To 12
    Cells(i + 1, 2) = (Cells(i + 1, 2) + Montant) * (1 + Taux)
Next i
```

3. Éventuellement, cliquer sur le bouton *Réinitialiser* ( ) pour arrêter l'exécution de la macro ou sur le bouton *Exécuter* (ou **F5**) pour poursuivre l'exécution de la macro automatiquement

LANCEMENT DU PAS À PAS DÉTAILLÉ À PARTIR DE L'ÉDITEUR DE VISUAL BASIC

1. Dans l'éditeur de Visual Basic (**Alt | F11**), cliquer dans la procédure voulue
2. Menu Débogage
   Pas à pas détaillé
3. Comme précédemment, appuyer sur la touche **F8** pour faire défiler les instructions une par une.

ARRÊTER UNE MACRO À UNE INSTRUCTION PRÉCISE

Il est possible de stopper une macro à la ligne de son choix, permettant ainsi, par exemple, de vérifier la valeur prise par une instruction à un moment précis de la procédure

EXÉCUTER UNE MACRO JUSQU'AU CURSEUR

Il est possible d'exécuter une macro jusqu'à la ligne contenant le curseur :

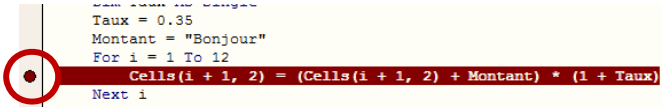
1. Dans l'éditeur de Visual Basic, cliquer n'importe où dans la ligne où s'arrêtera le programme
2. Menu Débogage
   Exécuter jusqu'au curseur (ou **Ctrl | F8**)

✓ La macro s'exécute jusqu'à la ligne avant celle choisie.

POINT D'ARRÊT

Le point d'arrêt permet d'arrêter l'exécution d'une macro à la ligne de son choix, permettant ainsi de voir le résultat de sa procédure à un moment précis :

1. Cliquer dans la marge de sélection, devant la ligne où sera arrêtée la macro
   ou menu Débogage, Basculer le point d'arrêt ou **F9** :



   En lançant la macro, celle-ci s'exécutera jusqu'à la ligne choisie (l'instruction sur fond rouge ne sera pas exécutée). Pour pouvoir exécuter la macro normalement :

2. Cliquer sur le bouton *Réinitialiser* ( )
3. Cliquer sur le point d'arrêt (le point rouge), pour le retirer.

✓ Faire **Ctrl | ⇧ | F9** pour retirer tous les points d'arrêt

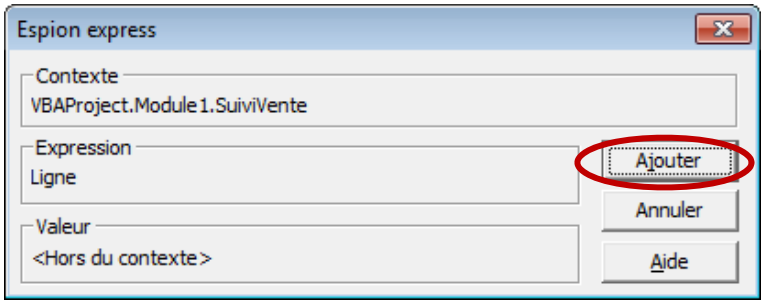
UTILISER DES ESPIONS POUR SURVEILLER LA VALEUR D'UNE VARIABLE

Il est possible de connaître la valeur d'une variable lorsqu'une macro se bloque avant la fin, en créant un espion chargé de surveiller la valeur prise par cette variable.

POUR SURVEILLER UNE VARIABLE AVEC L'ESPION EXPRESS

L'espion express permet de définir rapidement la variable à surveiller :

1. Dans la procédure, sélectionner la variable à surveiller
2. Menu Débogage
   Espion express (ou **⇧ | F9**)



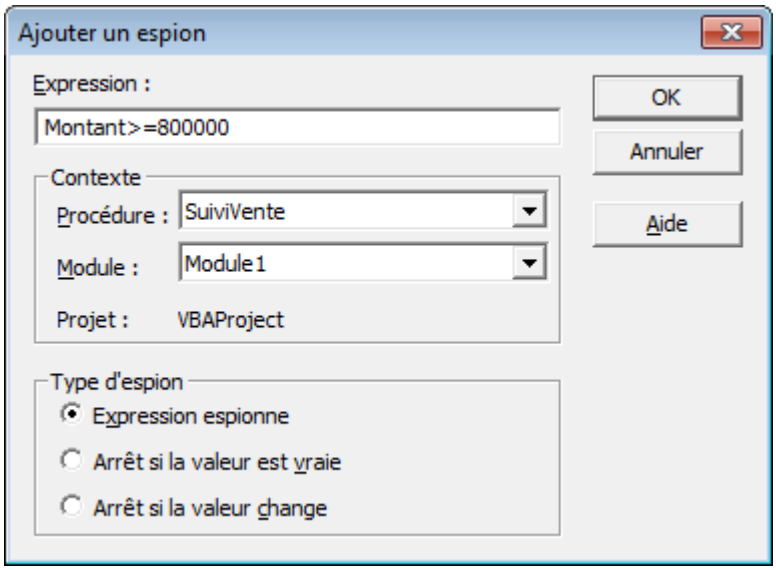
3. Cliquer sur **Ajouter**

POUR SURVEILLER UNE EXPRESSION AVEC L'AJOUT D'ESPION

Il est également possible de surveiller une expression (exemple : Montant >= 800000) et, éventuellement, d'arrêter la macro lorsque celle-ci est vraie :

1. Cliquer dans la procédure voulue
2. Menu Débogage
   Ajouter un espion

3. Dans la fenêtre qui apparaît, choisir les options voulues :

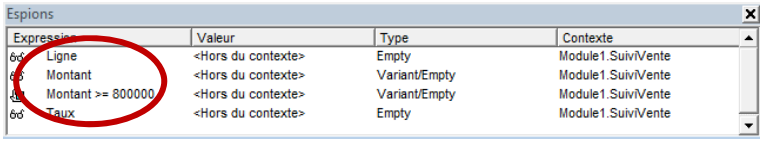


- **Expression** : saisir la variable ou l'expression voulue, exemple : Montant >= 800000
- **Procédure** : nom de la procédure en cours
- **Module** : nom du module en cours
- **Expression espionne** : affiche la valeur obtenue si la macro se bloque
- **Arrêt si la valeur est vraie** : arrête la macro si la valeur de la variable est vraie
- **Arrêt si la valeur change** : arrête la macro si la valeur de la variable change

4. Et **OK**

POUR SUPPRIMER UNE VARIABLE

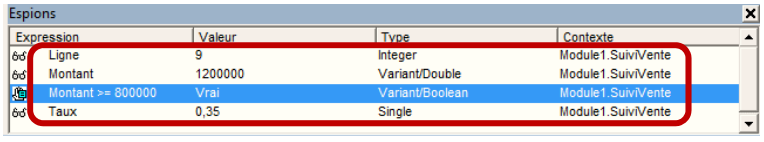
1. La fenêtre *Espions* apparaît avec les différentes variables surveillées



2. Dans cette fenêtre, faire un clic droit sur la variable à supprimer et cliquer sur *Supprimer un espion*

RETROUVER LES VALEURS PRISES PAR LA VARIABLE

1. Exécuter la macro normalement
2. Si celle-ci se bloque à cause d'une erreur ou d'une demande d'arrêt, les valeurs et le type des variables surveillées apparaîtront dans la fenêtre espion :



✓ Penser à réinitialiser la macro avec le bouton *Réinitialiser* ( ) pour pouvoir la relancer

FENÊTRES DE SUIVI DES MACROS

Plusieurs fenêtres permettent d'afficher les valeurs prises par les variables au moment où la macro se bloque. Pour activer ses fenêtres :

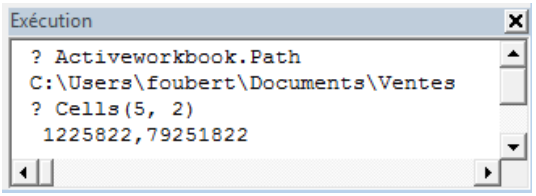
Menu Affichage

Cliquer sur la fenêtre voulu : Exécution, Variables locales, Espions ou Pile des appels

Fenêtre Exécution

La fenêtre *Exécution* permet de tester la valeur prise par une variable ou une expression :

Saisir l'expression voulue en la faisant précéder d'un point d'interrogation et en la validant en appuyant sur la touche **Entrée**

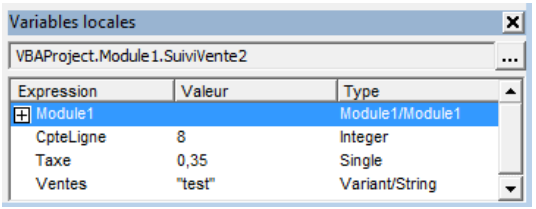


Exemples

- ? Activeworkbook.Path pour afficher le chemin actuel du fichier
- ? Cells(5, 2) pour connaître le contenu de la cellule B5

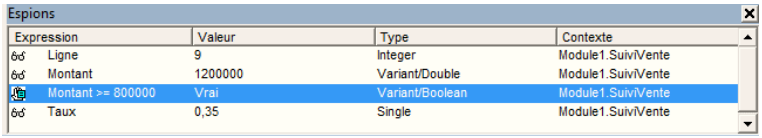
Fenêtre Variables locales

Lorsque la fenêtre variable locale est activée, elle affiche le contenu de chacune des variables de la procédure, si celle-ci s'interrompt avant la fin :



Fenêtre Espions

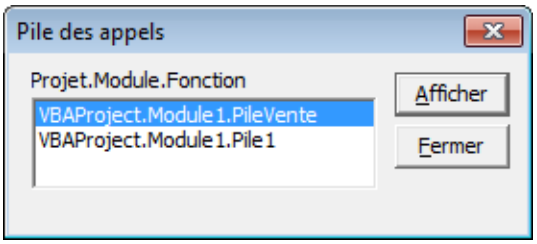
La fenêtre *Espions* est celle qui affiche les valeurs des variables ou des expressions surveillées



Fenêtre Pile des appels

La fenêtre *Pile des appels* permet de voir quelle procédure a appelé une autre procédure

1. Dans la fenêtre des piles d'appels, cliquer sur la procédure voulue



2. Cliquer sur le bouton **Afficher**, l'instruction qui a appelé une autre procédure— ou la ligne d'arrêt du code — s'affiche automatiquement, ce qui permet de retrouver d'où vient une erreur.

✓ Cette fenêtre ne peut être ouverte qu'après l'arrêt d'un programme en mode débogage.

VÉRIFIER LE TYPE DE DONNÉE D'UNE VARIABLE OU D'UNE CELLULE

En cas d'erreur, il peut être intéressant de vérifier le type (nombre, texte, date, etc.) d'une variable ou d'une cellule. Il existe des fonctions permettant de vérifier le type d'une variable ou d'une cellule. Celles-ci renvoient une valeur Vrai (*True*) ou Faux (*False*).

VÉRIFICATION QUE LA CELLULE ACTIVE CONTIENT BIEN UN NOMBRE

Voici un exemple de code avec *IsNumeric* où une boîte de dialogue apparaît si la cellule active contient un nombre :

```
If IsNumeric(ActiveCell) Then
    MsgBox "Cette cellule contient une valeur numérique"
Else
    MsgBox "Cette cellule ne contient pas un nombre"
End If
```

L'ACTION EST DÉCLENCHÉE SI CE N'EST PAS UNE ERREUR

Si la variable *AcCh* ne contient pas un message d'erreur, alors un message est affiché :

```
If Not IsError(AcCh) Then
    MsgBox "Aucune erreur détéctée!"
End If
```

QUELQUES FONCTIONS DE VÉRIFICATION DU TYPE D'UN ARGUMENT

IsDate() : Vrai si l'argument est une date

IsEmpty() : Vrai si l'argument est vide

IsError() : vrai si l'argument renvoie une erreur

POUR CONNAÎTRE LE TYPE D'UN ARGUMENT

TypeName renvoie une valeur indiquant le type de l'objet :

```
MsgBox TypeName(ActiveCell)
```

QUELQUES VALEURS RENVOYÉES PAR *TYPENAME*

Valeur renvoyée	Variable
Byte	Octet
Integer	Entier
Long	Entier long
Single	Nombre à virgule flottante en simple précision
Double	Nombre à virgule flottante en double précision
Currency	Monétaire
Decimal	Décimale
Date	Valeur de date
String	Chaîne de texte
Boolean	Valeur booléenne
Error	Valeur d'erreur
Empty	Non initialisée
Null	Aucune donnée valide
Object	Objet

ACCÈS À UNE PARTIE D'UNE MACRO EN CAS D'ERREUR

L'instruction *On Error* permet de déclencher une action en cas d'erreur — atteindre une ligne particulière de la procédure, ne pas exécuter la ligne erronée, etc. — sans qu'un message d'erreur n'apparaisse à l'écran.

ON ERROR GOTO

L'instruction *On Error Goto* permet d'atteindre une ligne spécifique en cas d'erreur

EXEMPLE

Si le classeur *gestion.xlsx* n'est pas ouvert, la macro ouvre le classeur avant de l'activer en allant à l'étiquette « ErreurOuverture » :

```
Sub AtteindreClasseur()
    ' En cas d'erreur (le classeur n'est pas ouvert lors de son activation)
    ' la macro va à l'étiquette ErreurOuverture: pour ouvrir le classeur
    On Error GoTo ErreurOuverture
    Workbooks("gestion.xlsx").Activate ' Activation du classeur
    On Error GoTo 0 ' Arrêt de la gestion des erreurs
    MsgBox "Le classeur a été activé"
    Exit Sub ' arrêt de la macro
' Ouverture du classeur en cas d'erreur (s'il n'est pas ouvert)
ErreurOuverture:
    Workbooks.Open ThisWorkbook.Path & "\gestion.xlsx" ' Ouverture du classeur
    MsgBox "Le classeur a été ouvert"
    Resume ' Retour à la ligne suivant la ligne erronée
End Sub
```

ON ERROR GOTO 0

L'instruction *On Error Goto 0* annule la précédente instruction *On Error Goto…* La macro continuera donc de se dérouler normalement et s'arrêtera en cas d'erreur, sans retourner à l'étiquette « ErreurOuverture »

ON ERROR RESUME NEXT

L'instruction *On Error Resume Next* permet de poursuivre la procédure en cas d'erreur, sans exécuter la ligne erronée.

```
Sub Vente()
    Dim i As Integer
    Dim Taux As Single
    On Error Resume Next
    Taux = 0.35
    Montant = "Bonjour" ' Montant contient du texte au lieu d'un nombre
    For i = 1 To 12
        Cells(i + 1, 2) = (Cells(i + 1, 2) + Montant) * (1 + Taux)
    Next i
End Sub
```

Normalement, la ligne avec `Cells(i + 1, 2) =…` devrait bloquer le programme car le champ *Montant* contient du texte qui ne peut être additionné. Toutefois, la commande On Error Resume Next empêche l'arrêt du programme. Celui-ci s'exécute donc jusqu'au bout mais il sera peu aisé de savoir pourquoi les cellules du tableau ne sont pas modifiées.

✓ Cette instruction est donc déconseillée car elle ne permet pas de situer — et donc de corriger — une erreur dans une procédure.

INFORMATIONS SUR LES ERREURS D'EXÉCUTIONS

L'objet *Err* stocke des informations permettant d'identifier une erreur

ERR.NUMBER ET ERR.DESCRIPTION

Err.Number : affiche le numéro de l'erreur

Err.Description : affiche la description de l'erreur

EXEMPLE

Err.Number affiche le numéro de l'erreur (13) et **Err.Description** la description de l'erreur (Incompatibilité de type) à cause de la variable *Montant* qui contient du texte au lieu d'un nombre.

L'instruction On Error Resume Next évite que la macro s'arrête à la ligne erronée.

```
Sub MsgErreur()
    Dim Montant As String
    On Error Resume Next
    Montant = "Bonjour"
    Montant = Montant + 10
    MsgBox Err.Number & Chr(10) & Err.Description
End Sub
```

✓ Les informations fournies par *Err.Number* et par *Err.Description* sont les mêmes que celles du message d'erreur du Visual Basic.

ERR.CLEAR

La fonction **Err.Clear** permet de remettre à zéro le message d'erreur

EXEMPLE

Le programme suivant parcourt les cellules de B2 à B12 — Cells(i + 1, 2) — et augmente la valeur de la cellule de 25%.

Toutefois, si la cellule contient du texte au lieu d'un nombre, le programme se poursuit (grâce à *On Error Resume Next*) mais affiche un message d'erreur (If Err.Number <> 0 Then…) pour chaque cellule ayant du texte :

```
Sub EffaceErreur()
    Dim i As Integer
    On Error Resume Next
    For i = 1 To 12
        Cells(i + 1, 2) = Cells(i + 1, 2) * 1.25 ' Augmente la valeur de 25%
        If Err.Number <> 0 Then ' Ne s'exécute qu'en cas d'erreur
            MsgBox "Attention, il y a une erreur"
            Err.Clear ' Efface le contenu de Err
        End If
    Next i
End Sub
```

✓ Sans *Err.Clear*, *Err.Number* resterai toujours égal à 13 (pour incompatibilité de type) et donc, le message d'erreur s'afficherai systématiquement, même si le contenu de la cellule est un nombre.

LIENS SUR LA GESTION DES ERREURS

Présentation des différentes instructions de gestion des erreurs :

https://silkyroad.developpez.com/VBA/GestionErreurs

Débogae des erreurs :

https://cafeine.developpez.com/access/tutoriel/debugprint