Ecrire ou modifier une macro

Il est rare qu'une macro générée automatiquement donne entièrement satisfaction. On ne peut, par ce moyen qu'effectuer des opérations mathématiques ou logiques sur les données d'un tableau ; il n'est pas possible de faire des calculs intermédiaires, par exemple.
Par ailleurs, Excel ne génère pas de déclaration de données et travaillera avec le format qui lui semblera le plus approprié et qui ne sera pas forcément celui que vous auriez choisi.

Il est donc nécessaire de pouvoir modifier une macro ou de l'écrire entièrement à la main.

1 - Début et fin de la procédure

Comme il peut y avoir plusieurs procédures dans un module, il est nécessaire de délimiter le début et la fin de chacune.

Une procédure doit toujours commencer par :

Sub Macro()

Nom de la macro

Parenthèses gauche et droite

Une procédure se termine toujours par :

End Sub

Ces instructions doivent être seules sur leur ligne :

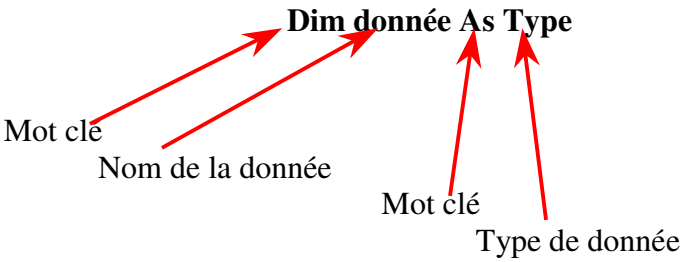
```
Sub MacroDémo()
Déclarations des données
Instructions
……
End Sub
```

2 - Déclarer les données de travail

Rappel : Les données de travail doivent être déclarées :
Soit en début de module pour les données communes à plusieurs macros
Soit en début de macro pour les données spécifiques à le macro.

Les données à déclarer sont toutes celles qui vont être nécessaires pour mémoriser des valeurs intermédiaires au cours de nos calculs.

La syntaxe de déclaration des données est :



Les différents types de données les plus courants sont les suivants :

Type	Caractéristiques
Boolean	Les variables de type Boolean sont stockées sous la forme de nombres de 16 bits (2 octets), mais elles ne peuvent avoir pour valeur que **True** ou **False**. Elles s'affichent sous la forme `True` et `False` (avec l'instruction **Print**) ou `#TRUE#` et `#FALSE#` (avec l'instruction **Write #**). Utilisez les mots clés **True** et **False** pour faire passer d'un état à l'autre des variables de type **Boolean**.Lorsque d'autres types de données numériques sont convertis en valeurs de type **Boolean**, 0 devient **False** et toutes les autres valeurs deviennent **True**. Lorsque des valeurs de type **Boolean** sont converties en d'autres types de données, **False** devient 0 et **True** devient -1.
Byte	Les variables de type Byte sont stockées sous la forme d'un nombre de 8 bits (1 octet unique), non signé, compris entre 0 et 255.Le type de données Byte est utile pour le stockage de données binaires.
Currency	Les variables de type Currency sont stockées sous la forme de nombres de 64 bits (8 octets) au format entier, avec un décalage de 10 000 afin d'obtenir un nombre à virgule fixe comprenant 15 chiffres à gauche du séparateur décimal et 4 chiffres à droite. Cette représentation offre une plage comprise entre -922 337 203 685 477,5808 et 922 337 203 685 477,5807. Le type de données Currency est utile pour les calculs monétaires et pour les calculs à virgule fixe dans lesquels la précision revêt une importance particulière.

Date	Les variables de type Date sont stockées sous la forme de nombres à virgule flottante de 64 bits (8 octets) IEEE représentant des dates comprises entre le 1er janvier 100 et le 31 décembre 9999, et des heures allant de 0:00:00 à 23:59:59. Toute valeur de littéral date peut être attribuée à une variable de type **Date**. Les littéraux date doivent être délimités par le signe **#**, par exemple `#January 1, 1993#` ou `#1 Jan 93#`. Les variables de type **Date** affichent les dates au format de date abrégé reconnu par votre ordinateur. Les heures s'affichent au format horaire (plage de 12 ou 24 heures) défini dans votre ordinateur. Lorsque d'autres types de données numériques sont convertis en données de type **Date**, les valeurs situées à gauche du séparateur décimal représentent la date, tandis que celles situées à droite correspondent à l'heure. Minuit est représenté par 0 et midi par 0,5. Les nombres entiers négatifs représentent des dates antérieures au 30 décembre 1899.
Double	Les variables de type Double (à virgule flottante en double précision) sont stockées sous la forme de nombres à virgule flottante de 64 bits (8 octets) IEEE dont la valeur est comprise entre -1,79769313486231E308 et -4,94065645841247E-324 pour les nombres négatifs et entre 4,94065645841247E-324 et 1,79769313486231E308 pour les positifs.
Integer	Les variables de type Integer sont stockées sous la forme de nombres de 16 bits (2 octets) dont la valeur est comprise entre -32 768 et 32 767. Le caractère de déclaration de type **Integer** est le signe **%**. Les variables de type **Integer** permettent également de représenter des valeurs énumérées. Celles-ci peuvent contenir un ensemble fini d'entiers uniques possédant tous une signification particulière dans le contexte où ils sont utilisés. Elles permettent d'opérer facilement une sélection parmi un nombre connu de choix, du type noir = 0, blanc = 1, etc. Il est conseillé de définir des constantes pour chaque valeur énumérée via l'instruction **Const**.
Long	Les variables de type Long (entier long) sont stockées sous la forme de nombres signés de 32 bits (4 octets) dont la valeur est comprise entre -2 147 483 648 et 2 147 483 647.
Object	Les variables de type Object sont stockées sous la forme d'adresses 32 bits (4 octets) qui font référence à des objets. L'instruction **Set** permet d'affecter une référence d'objet à une variable déclarée comme **Object**. **Note** : Une variable déclarée comme **Object** est suffisamment flexible pour contenir une référence à n'importe quel type d'objet, mais la liaison à l'objet désigné par la variable est toujours tardif (liaison au moment de l'exécution). Pour obtenir une liaison précoce (liaison au moment de la compilation), attribuez la référence d'objet à une variable déclarée avec un nom de classe spécifique.
Single	Les variables de type Single (à virgule flottante en simple précision) sont stockées sous la forme de nombres à virgule flottante de 32 bits (4 octets) IEEE dont la valeur est comprise entre -3,402823E38 et -1,401298E-45 pour les nombres négatifs et entre 1,401298E-45 et 3,402823E38 pour les positifs.
String	Chaîne de caractères.

Dans la pratique, les types de données les plus utilisés sont :

Integer pour les calculs
String pour les chaînes de caractères.

3 - Ecrire les instructions

Remarque préliminaire : je vous conseille de toujours commencer par faire exécuter une opération à l'aide de l'enregistreur automatique de macro. Ceci vous donnera une structure de programme pré écrite qu'il n'y aura plus qu'à modifier manuellement. Faites ceci pas à pas pour des "morceaux" de votre macro que vous pourrez ainsi construire progressivement.

Les syntaxes d'instructions sont écrites en bleu.

Une série (bloc) d'instructions doit toujours :

- lire dans une feuille Excel la ou les données à traiter,
- effectuer le traitement choisi : logique, traitement de texte calcul ….
- fournir le résultat là où il est attendu.

3.1 - Ouvrir un fichier Excel

Si les données à traiter figurent dans un autre fichier Excel que celui contenant la macro, vous pouvez en demander l'ouverture automatique.

```
Workbooks.Open Filename:="C:\Mesdocs\Sudoku.xls"
```

Le nom de fichier doit être écrit, soit en toutes lettres et entre guillemets en spécifiant le chemin d'accès complet, comme ci-dessus, soit à partir d'une zone de travail pré remplie avec ce nom :

```
Workbooks.Open Filename:=Nomfichier
```

Dans ce cas, il faut au préalable :

- déclarer une variable appelée Nomfichier, par exemple, et de type String
- lire dans la feuille Excel le nom du fichier et le mémoriser dans cette variable.

3.2 - Sélectionner et lire dans une feuille Excel

Comme dans tout logiciel, avant de faire une action, il faut dire "OU" avant de dire "QUOI". Il est donc nécessaire d'indiquer à Excel sur quelle zone du tableau on veut agir.

On peut sélectionner une seule cellule ou une plage contiguë.

Pour sélectionner *une seule cellule*, par exemple la cellule B25, il faut écrire :

```
Windows("Sudoku.xls").Activate   Nom du classeur Excel (si différent du classeur en cours)
Sheets("Vierges").Select         Nom de la feuille dans le classeur.
Range("B25").Select              Emplacement de la cellule sélectionnée.
```

Pour sélectionner *une plage de cellules contiguës dont on connaît* précisément les coordonnées, par exemple, de A1 à B25 :

```
Range("A1:B25").Select
```
Bornes de la plage sélectionnée.

Pour sélectionner *une plage de cellules contiguës dont on ne connaît* pas précisément les coordonnées, celles-ci étant déterminées dans une autre phase du programme :

```
Range(Cells(Lignedeb, n), Cells(Lignefin, m)).Select
```

N° de la ligne de début de sélection
N° de la colonne de début de la sélection
N° de la ligne de fin de la sélection
N° de la colonne de fin de la sélection

Attention : dans cette syntaxe de rédaction, on ne raisonne plus par des lettres pour les colonnes et des chiffres pour les lignes, mais directement par des numéros dans les deux cas. On peut indiquer ces numéros, soit en clair, soit par référence à des zones de travail qui contiennent ces valeurs.

Lorsque la sélection est effectuée, on peut agir dessus : soit lire le contenu de la cellule, si on n'en a sélectionné qu'une seule, soit faire une action de mise en forme sur l'ensemble d'une plage.

Pour lire une cellule et mettre son contenu dans une zone de travail :

```
Zonetrav = ActiveCell
```

Le contenu de la cellule "active" (= sélectionnée) est transféré dans la zone de travail de la macro.

3.3 - Rechercher une valeur

On peut avoir besoin de rechercher une valeur dans une feuille, comme on le ferait manuellement. Cette valeur peut être n'importe où dans la feuille ou dans une colonne sélectionnée ou dans une plage de cellules sélectionnées.

```
Columns("E:E").Select
```
Sélection de la plage de recherche (ici, la colonne E)

```
On Error Resume Next
```
Traitement des erreurs : que fait on si on ne trouve pas la valeur, le mieux est d'aller à l'instruction suivante et de traiter le cas à la main, mais on peut insérer un débranchement vers un paragraphe de traitement des erreurs.

```
Selection.Find(What:=Reg1,
After:=ActiveCell, LookIn:=xlFormulas,
LookAt _
:=xlPart, SearchOrder:=xlByRows,
SearchDirection:=xlNext, MatchCase:= _
False).Activate
```
Formule de recherche:
What = ce que l'on recherche; si c'est une valeur fixe, il faut la mettre entre guillemets. (ici, on recherche une cellule de la colonne E contenant ce qui est dans la variable Reg1.

L'instruction est longue et ne tient pas sur une seule ligne. Il faudra l'écrire de la façon suivante :

```
Selection.Find(What:=Reg1, After:=ActiveCell, LookIn:=xlFormulas, LookAt _
    :=xlPart, SearchOrder:=xlByRows, SearchDirection:=xlNext, MatchCase:= _
    False).Activate
```

Pour indiquer à Excel que l'instruction se continue sur la ligne suivante, il faut terminer la ligne par un espace et un blanc souligné (underscore, sous le 8)

3.4 - Trier des valeurs

On peut avoir besoin de trier une plage de valeurs, comme on le ferait manuellement (le tri se fait toujours verticalement dans une colonne) :

`Range("A3").Select`	Colonne du premier critère de tri : c'est la première cellule de la colonne sous les lignes de titres; les lignes au dessus ne sont pas prises en compte si la fenêtre a été séparée avec des volets figés.

`Selection.Sort Key1:=Range("A3"),`
`Order1:=xlAscending, Key2:=Range("B3") _`
`, Order2:=xlAscending, Key3:=Range("D3"),`
`Order3:=xlAscending, Header:= _`
`xlGuess, OrderCustom:=1,`
`MatchCase:=False,`
`Orientation:=xlTopToBottom`
	Premier critère : colonne A, tri ascendant
	Deuxième critère : colonne B, tri ascendant
	Troisième critère : colonne D, tri ascendant.
	(on peut ne mettre qu'un ou 2 critères de tri).
	
	Tri du haut vers le bas.

Comme ci-dessus, l'instruction devra être écrite de la façon suivante :

```
Selection.Sort Key1:=Range("A3"), Order1:=xlAscending, Key2:=Range("B3") _
    , Order2:=xlAscending, Key3:=Range("D3"), Order3:=xlAscending, Header:= _
    xlGuess, OrderCustom:=1, MatchCase:=False, Orientation:=xlTopToBottom
```

3.5 - Calculer

On ne peut pas calculer directement dans la feuille Excel. Il est nécessaire de rapatrier les valeurs dans les zones de travail (voir §19.7.3.2) puis de faire les calculs sur ces zones de travail.
Les zones de travail servant aux calculs doivent avoir été déclarées comme des zones numériques. Exemple :

```
Dim Compteur1 As Integer
Dim Compteur2 As Integer
```

Au lancement de la macro, toutes ces zones sont initialisées à 0 (zéro).
Sur ces zones on peut faire tous les calculs nécessaires.

Initialiser un compteur avec une valeur donnée : Compteur1 = 2
Additionner deux compteurs : Compteur1 = Compteur1 + Compteur2
(le résultat est dans Compteur1)

Compteur3 = Compteur1 + Compteur2
(le résultat est dans Compteur3)
Multiplier : Compteur1 = Compteur2 * 25
Compteur3 = Compteur1 * Compteur2
Soustraire : Compteur1 = Compteur1 - 3
Diviser : Compteur1 = Compteur2 / 4

Etc.

Le résultat est toujours dans la zone de travail indiquée à gauche du signe =

3.6 - Traiter les chaînes de caractères

On ne peut pas manipuler une chaîne de caractères directement dans la feuille Excel. Il est nécessaire de rapatrier les valeurs dans les zones de travail (voir §19.7.3.2) puis de faire les opérations sur ces zones de travail.
Les zones de travail servant aux calculs doivent avoir été déclarées comme des zones "chaîne de caratère". Exemple :

```
Dim Chaine1 As String
Dim Chaine2 As String
```

Au lancement de la macro, toutes ces zones sont initialisées à vide.

On peut faire diverses opérations de traitement de chaîne :

Affecter une valeur à une chaîne : Chaine1 = "valeurs de départ"
Chaine1 = Chaine2
Si la valeur indiquée est littérale, elle doit être entre guillemets.

Concaténer plusieurs chaînes : Chaine1 = Chaine2 & Chaine3 & "suite"
Le signe "&" indique que les chaînes seront mises bout à bout. Si on veut intégrer un texte en clair dans cette opération, il faut le mettre entre guillemets

Extraire les n caractères de gauche de la chaîne : Chaine2 = Left(Chaine1, n)
La zone Chaine2 contiendra les n caractères de gauche de la zone Chaine1

Extraire les n caractères de droite de la chaîne : Chaine2 = Right(Chaine1, n)
La zone Chaine2 contiendra les n caractères de droite de la zone Chaine1

Remarque : une chaîne de caractères ne peut pas contenir plus de 256 caractères.

3.7 - Les tests et les étiquettes de débranchement

Il est parfois nécessaire de tester la valeur d'une zone (de travail ou d'une cellule) et de faire un traitement particulier ou un autre suivant la valeur trouvée. Exemples :

Débranchements :

If Zonetrav = "xxx" Then GoTo Suite	Test de la zone "Zonetrav". Si elle est égale à "xxx", aller directement à l'étiquette "Suite".
If ActiveCell = "xxx" Then GoTo Suite	Idem avec le contenu de la cellule sélectionnée

Une étiquette doit être en début de ligne et être suivie de :

```
Suite:
```

Traitement lié à la condition :
On peut avoir intérêt à écrire directement après le "If", les quelques lignes de traitement à effectuer. Il faut dire à Excel où s'arrête le bloc d'instructions lié à la condition :

```
If Zonetrav = "xxxx" Then
        Instruction 1
        Instruction 2
        …..
        End If
```

3.8 - Les boucles logiques

De la même façon que précédemment, on peut écrire des boucles de traitement avec débranchement, ou avec un traitement effectué tant qu'une condition est remplie.

Boucle avec débranchement :

```
Débutboucle:
        Test manuel de condition de fin de boucle pour en sortir
        Instruction
        Instruction
        ….
        GoTo Debutboucle
```

Boucle avec traitement lié à une condition (= faire, tant que) :

```
        Do While essai2 < essai1
        essai2 = essai2 + 1
        autres instructions
        Loop
```

Toutes les instructions se trouvant entre "Do" et "Loop" seront exécutées autant de fois que nécessaire, tant que la condition indiquée est remplie. Loop indique un branchement à la première instruction de la boucle.

3.9 - Le copier/couper - coller

Le copier/coller ou couper/coller peut être programmé dans une macro et concerner une seule cellule ou une plage.

```
Range(…..).Select                                   Sélection de la plage à copier
Selection.Copy                                      Copie
Range(…..).Select                                   Positionnement pour le coller
ActiveSheet.Paste                                   Action de coller (standard)
```

Ou :

```
Range(…….).Select                                   Sélection de la plage à copier
Selection.Copy                                      Copie
Range(…).Select                                     Positionnement pour le coller
Selection.PasteSpecial Paste:=xlValues, Operation:=xlNone, SkipBlanks:= _
    False, Transpose:=False                         Action de coller (collage spécial)
```

Pour faire un couper, remplacer "Copy" par "Cut".

3.10 - Réécrire le résultat dans la feuille Excel

L'objectif de la macro est au final de fournir un résultat lisible dans la feuille Excel ! Il faut donc faire l'opération inverse de la lecture vue ci-dessus.

```
Range("B25").Select                 Sélection de la cellule devant contenir le résultat
ActiveCell.FormulaR1C1 = Zonetrav   Recopie dans la cellule sélectionnée du contenu
                                    De la zone de travail "Zonetrav".
```

4 - Insérer des commentaires

Comme vous l'imaginez, la syntaxe du langage Visual Basic pour Excel est assez abstraite. De plus, on a du mal à mémoriser certaines logiques employées pou réaliser telle ou telle fontion.

Il est donc nécessaire d'insérer dans le programme des commentaires permettant d'expliciter telle ou telle instruction ou la logique d'un pavé d'instructions.

Pour insérer un commentaire, commencez une nouvelle ligne par le caractère ' (sous le 4) et tapez librement votre commentaire. La ligne apparaîtra en vert et ne sera pas traitée comme une instruction :

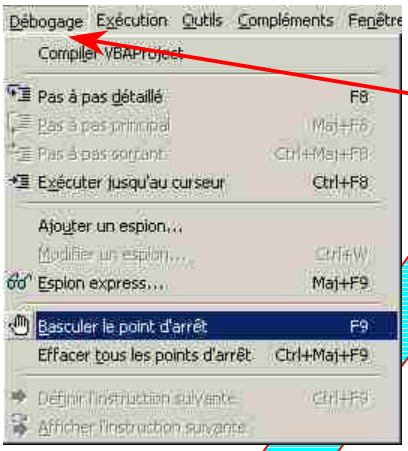
```
……
Range("A1").Select
'
' Elaboration du critère de recherche dans la bonne feuille d'organigramme
' Mise en mémoire des éléments
'
Sheets("Adresses(L)").Select
Let Numligne = ActiveCell.Row
……
```

De même, pour aérer vos programmes, vous pouvez insérer des lignes vides qui ne sont pas prises en compte et qui ne gênent pas le déroulement des procédures.

5 - Aides à la mise au point - Déboguage

Il est "rare" qu'une procédure soit écrite correctement du premier coup. Il est nécessaire de faire des tests pour vérifier son fonctionnement correct.

Visual Basic offre un certain nombre d'outils d'aide au "déboguage" et à la mise au point.



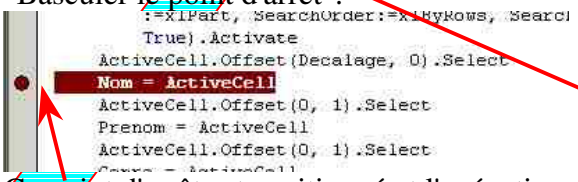
Ces outils sont regroupés dans le menu :

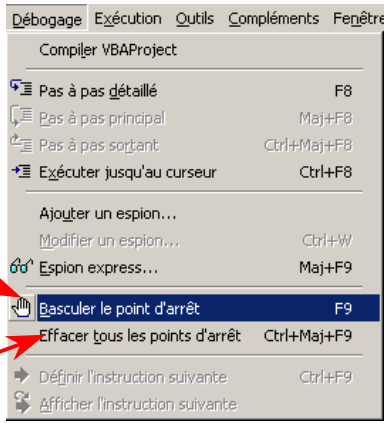
"Déboguage" de Visual Basic

.

5.1 - Création d'un point d'arrêt

La création d'un point d'arrêt se fait en positionnant le curseur à un endroit précis de la procédure et en cliquant sur "Basculer le point d'arrêt".





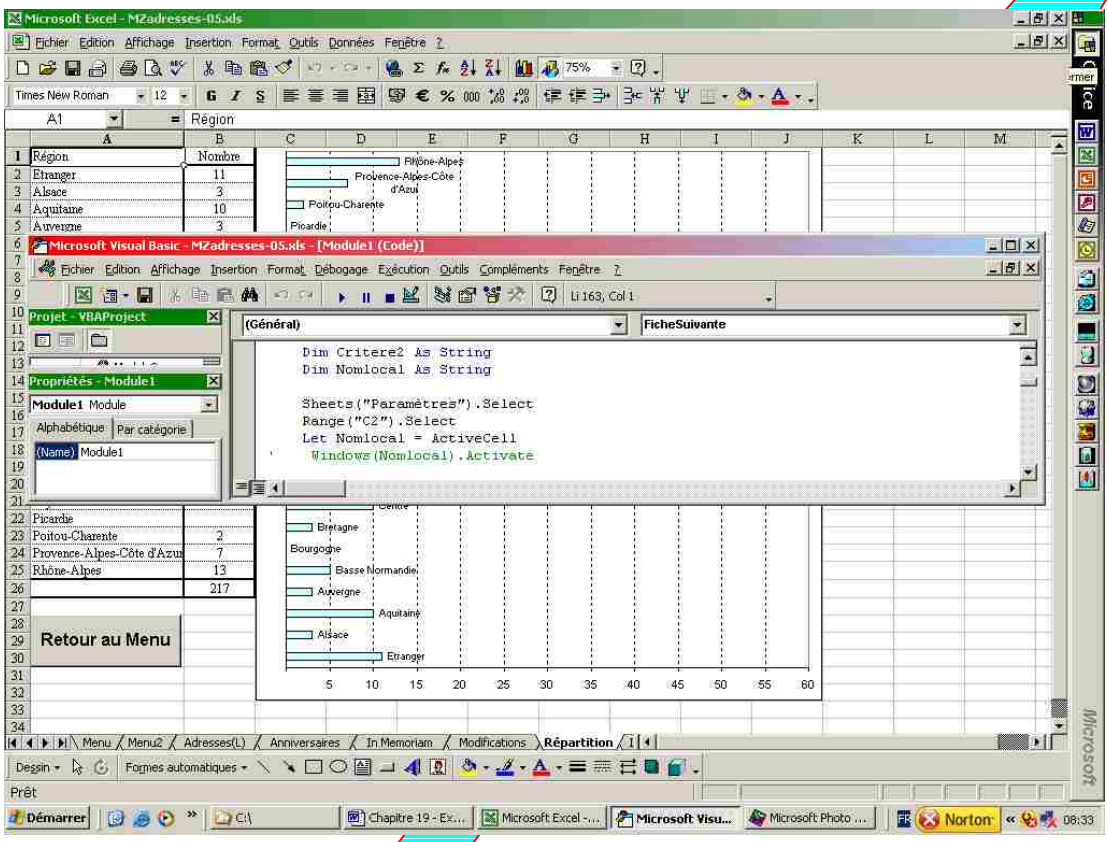
Ce point d'arrêt est positionné et l'exécution de la procédure s'arrêtera à cet endroit.
Lorsque la procédure est au point, il faut

effacer ces points d'arrêt.

5.2 - Exécution pas à pas détaillée

L'exécution pas à pas permet d'exécuter les instructions une par une et de vérifier l'action effectuée.

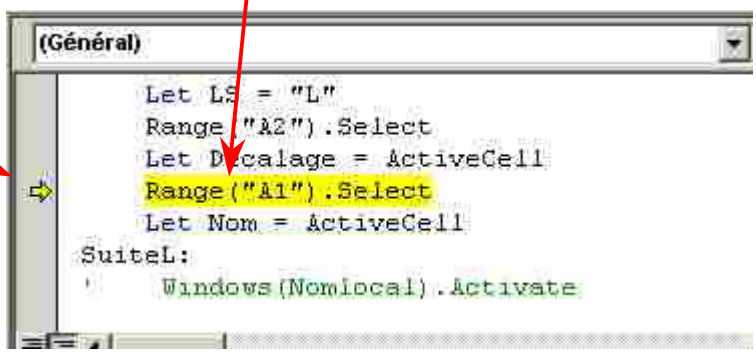
Réduisez la taille de Visual Basic et superposez le à la feuille Excel :



Vous pourrez ainsi contrôler simultanément le déroulement de la procédure et son action dans la ou les feuilles Excel concernées.

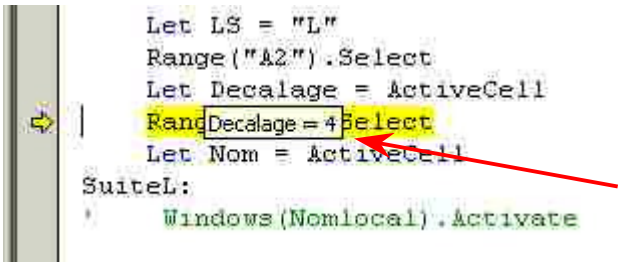
Déclenchez l'exécution pas à pas à l'aide de la touche de fonction **"F8".**

L'instruction qui va être exécutée passe en sur lignage jaune avec une flèche sur la gauche.



Vous pouvez déplacer la flèche vers le haut ou vers le bas pour ré exécuter une ligne après modification ou pour sauter une instruction : cliquez sur la flèche jaune et maintenez le clic gauche appuyé, puis remontez ou descendez cette flèche (attention : le positionnement de la cellule sélectionnée dans la feuille Excel ne sera peut-être plus correct).

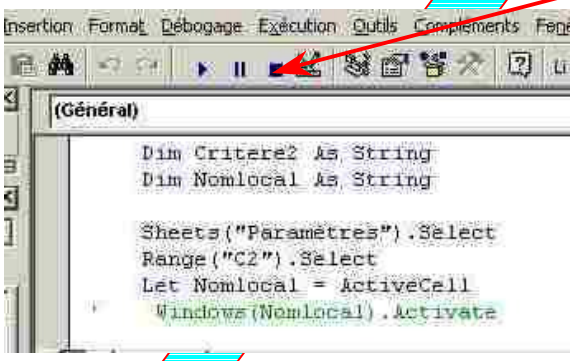
Lorsque vous êtes en mode pas à pas, vous pouvez vérifier à tout moment le contenu d'une variable.



Positionnez la souris au dessus de la variable dont vous voulez connaître la valeur (ici : Decalage) et laissez la immobile 1 ou 2 secondes.
La valeur contenue apparaît automatiquement (ici, la variable de travail Decalage contient le nombre 4).

Pour passer à l'instruction suivante, ré appuyez sur la touche **"F8".**

Pour arrêter l'exécution (pour corriger une erreur, par exemple), cliquez sur le bouton d'arrêt : (carré noir, comme sur votre lecteur de DVD ou votre magnétoscope).



5.3 - Positionnement d'espions.

Le positionnement d'espions servait dans les anciennes versions de Visual Basic pour afficher dans une fenêtre spécialisée le contenu d'une ou plusieurs variables de travail. Il est désormais possible de connaître ces contenus en positionnant la souris au dessus de la variable, comme indiqué ci-dessus. Le positionnement d'espions est donc devenu de peu d'intérêt.