Licence Economie et Gestion
Outils Informatiques 2
Responsable : C. Hanen

Cours n° 1

Objectifs du cours : définir la notion de macro, analyser sa structure et montrer l'environnement de programmation.
Pages 1 à 6 des transparents de cours à lire.

Eléments complémentaires :

Voici une macro qui a enregistré la manipulation suivante : mettre =C24+1 dans la cellule D24, et enregistrer le document excel.
On commente ses instructions en vert :

```
Sub formulecellule()
```

une macro commence par le mot réservé Sub suivi du nom de la macro (que l'on déclare lors de l'enregistrement. Elle se termine par End Sub (voir plus bas). Elle est composée de lignes, parfois prolongées par un signe _ qui indique que l'on prolonge la ligne sur la ligne suivante. Chaque ligne correspond à une instruction. Les instructions sont exécutées une à une par la machine.

```
'
'
    ActiveCell.FormulaR1C1 = "=RC[-1]+1"
```

Il s'agit ici d'une instruction qui écrit la formule relative dans la cellule active. Elle fait appel à un objet prédéfini que l'on étudiera en fin de semestre. Le signe = est une instruction d'affectation que l'on introduira au cours n°2.

```
    Range("D24").Select
```

'Cette instruction appelle une macro prédéfinie qui sélectionne la cellule D24.

```
    ChDir "C:\Documents and Settings\Administrateur\Mes documents\outils2"
```

'Cette instruction appelle une macro prédéfinie Chdir qui change le dossier de référence pour l'enregistrement. Le chemin d'accès est en fait un paramètre de la macro Chdir. L'appel de macros sera étudié dans le chapitre sur les fonctions (pages 25 à 30 des transparents) , et sera utilisé dès le début avec les entrées-sorties (pages 11 et 12).

```
 ActiveWorkbook.SaveAs Filename:= _
"C:\Documents and Settings\Administrateur\Mes documents\outils2\toto.xls" _
        , FileFormat:=xlNormal, Password:="", WriteResPassword:="", _
        ReadOnlyRecommended:=False, CreateBackup:=False
```

On voit ici un exemple d'instruction qui est rédigée sur plusieurs lignes avec le signe_ en fin de ligne. C'est une macro prédéfinie avec paramètres qui réalise l'enregistrement du document.

```
End Sub
```

Une fois qu'une macro est écrite on peut en modifier directement le texte, et à l'exécution les opérations seront modifiées. Par exemple, en modifiant la macro ainsi (modifications en rouge), la macro enregistre le document dans le dossier essai de mes documents en ayant préalablement sélectionné la cellule D36.

```
Sub formulecellule()
    ActiveCell.FormulaR1C1 = "=RC[-1]+1"
    Range("D36").Select
    ChDir "C:\Documents and Settings\Administrateur\Mes documents\essai"
ActiveWorkbook.SaveAs Filename:= _
"C:\Documents and Settings\Administrateur\Mes documents\outils2\toto.xls" _
        , FileFormat:=xlNormal, Password:="", WriteResPassword:="", _
        ReadOnlyRecommended:=False, CreateBackup:=False
End Sub
```

On verra dans la suite du cours que l'on peut également faire des calculs et les mémoriser (pages 8 à 10 et 13 à 15 des transparents) interagir avec l'utilisateur (pages 111-12), exécuter des instructions selon certaines conditions (pages 16-20) ,et répéter des groupes d'instructions (pages 21-24).

L'environnement d'édition et d'exécution des macros (Alt F11) les mémorise dans des fichiers appelés modules qui sont attachés au documents, et permet à l'utilisateur d'écrire des macros et de les exécuter soit en totalité soit pas à pas (touche F8) ce qui permet d'en visualiser les résultats intermédiaires.

Il est également possible de mettre des points d'arrêt à certains endroits (touche F9) pour permettre, en phase de programmation de vérifier étape par étape que la macro fait bien ce qu'on veut qu'elle fasse. On utilise alors l'exécution standard (F5).

Pour pouvoir être exécutée par la machine, la macro est traduite en langage machine et stockée en mémoire. C'est la compilation. La compilation permet également de vérifier que le programmeur n'a pas fait d'erreur de syntaxe. On peut compiler une macro sans l'exécuter dans le menu Debogage.

On appelle Débogage la phase de conception d'un programme qui consiste à le mettre au point et à le tester grâce à l'environnement de programmation.

Le saviez-vous ?
Un bogue (en anglais bug) est un mot qui désigne une erreur lors de l'exécution d'un programme. Il correspond le plus souvent à une erreur de conception. Mais bug signifie aussi cafard. Ce nom à été donné car les premiers calculateurs américains fonctionnaient dans d'immenses salles avec des circuits électriques parfois à l'air libre. Lorsque des insectes se glissaient dans la salle, il arrivait fréquemment qu'ils provoquent des courts-circuits, et perturbent grandement l'exécution des programmes.

Cours n°2

Objectifs du cours : Premiers pas dans VBA. Notion de variable, de type, d'affectation et d'entrée-sortie.
Pages 7 à 12 des transparents de cours à lire.

Eléments complémentaires :

1) Variables, affectation, types

On commente d'abord sur les macros suivante la notion de **variable**, de **type** et **d'affectation** et de **constante** selon le type.

```
Sub affectation()
Dim a As Integer
Dim b As Byte
Dim c As String
Dim d As Double
```

La variable a est destinée à contenir un entier relatif, la variable b un nombre entier entre 0 et 255, la variable c une chaîne de caractères et la variable d un nombre décimal. Ces instructions sont déclaratives, c'est-à-dire utilisées par le compilateur pour réserver des espaces en mémoire pour les données correspondantes, et permettre de traduire les instructions opératoires suivantes.

```
a = -5
```

Cette instruction consiste à ranger la valeur -5 dans la case mémoire symbolisée par la lettre a. On dit qu'on affecte la valeur -5 à la variable a.

```
b = 68
c = "quelle valeur?"
```

De la même manière, ici la valeur 68 est affectée à la variable b, et la chaîne de caractères `quelle valeur?` est rangée dans la variable c. A noter que les constantes chaînes de caractères doivent être entre guillemets.

```
d = a / b
```

On a ici un premier exemple de calcul effectué par une instruction. Ici la machine commence par évaluer l'expression à droite du signe = puis range la valeur correspondante dans la case mémoire symbolisée par d.
Pour évaluer cette expression, la machine remplace les symboles a et b par les valeurs, à ce moment là, des variables correspondantes. Donc ici a vaut -5, b vaut 68 et donc la machine calcule -5/68 et affecte cette valeur à d.

```
End Sub
```

Le saviez-vous ?
Du fait de la représentation des nombres limitée par un certain nombre de bits, il est impossible de tous les représenter (Rappel : sur n bits on représente 2^n nombres) De ce fait, en particulier pour les nombres à virgule, tous les calculs fait par une machine sont potentiellement faux. En effet, si le résultat d'un calcul de deux nombres représentés n'est pas représentable, le résultat fourni effectivement est un nombre représentable « proche ». Cette proximité, dans le cas de très grands nombres peut conduire à des nombres assez éloignés.

L'absence de précaution à ce sujet lors d'une évolution de la représentation des nombres d'un ordinateur de vol de la fusée Ariane a conduit à un défaut majeur dans le calcul de sa trajectoire et a causé son explosion en vol en 1996.

On commente maintenant les possibilités de **visualisation de l'environnement** ainsi que le mode **d'évaluation des expresssions**.

```
Sub calculsenchaine()
Dim a As Integer
Dim b As Integer
a = -5
b = a + 12
```

Ici on a deux variables entières a et b, la première se voit affecter la valeur -5 comme précédemment, et la seconde une expression calculée à partir de la valeur courante de a, donc -5 à laquelle on ajoute 12. Ainsi, après cette instruction b vaut 7.
A noter qu'en exécutant cette macro pas à pas (F8) on peut visualiser la valeur de chaque variable après chaque instruction en laissant le pointeur de souris glisser sur le nom de la variable.

```
a = 6
```

Il faut remarquer que maintenant a change de valeur et contient 6. b par contre n'a pas changé de valeur, et contient toujours la valeur 7. Il faut bien comprendre qu'une instruction d'affectation comme b=a+12 ne lie pas du tout les valeurs de a et de b comme pourrait le faire une formule excel. Il s'agit à chaque fois d'une évaluation instantanée des expressions par rapport aux valeurs qu'ont les variables à ce moment là.

```
End Sub
```

On commente enfin les problèmes liées aux **incompatibilités de types** dans les expressions et/ou aux **conversions automatiques de type**.

```
Sub calculfaux()
Dim a As Integer
Dim b As Byte
Dim c As String
Dim d As Double
a = -5
b = 68
c = "quelle valeur?"
d = c+a
```

Cette instruction va provoquer une erreur d'exécution, avec pour message « incompatibilité de type ». En effet, elle consiste à demander à la machine d'évaluer l'expression c+a. Or a est un nombre entier, et c une chaîne de caractères. Le signe + a deux sens possibles en VBA. Pour les nombres il s'agit de l'addition habituelle. Lorsqu'il s'agit de chaînes de caractères cela désigne la concaténation (comme l'opérateur &). Ici, la machine va convertir automatiquement l'entier a en une chaîne de caractères et calculer la chaîne

```
"quelle valeur?-5"
```

Ensuite, la machine va chercher à ranger cette valeur dans la variable d. Mais le type de celle-ci ne peut accueillir que des nombres. Il y a donc incompatibilité de type.

```
End Sub
```

Le saviez-vous ?
L'ordinateur fait beaucoup de choses pour vous. En particulier il a tendance à faire des conversions de type de manière automatique.
Par exemple, si a est une variable de type double qui vaut 3,6 et si b est une variable de type Integer, l'affectation b=a ne provoquera pas d'erreur d'exécution pour incompatibilité de type.
Par contre, la machine convertira automatiquement a en un entier – sans vous dire comment, probablement en tronquant sa partie fractionnaire (b vaudra alors 3).

Il est très important pour un programmeur de bien maîtriser les types de ses variables, afin d'éviter de laisser la machine faire ce qu'elle veut à votre insu avec les données.

Le langage VBA permet en fait de ne pas déclarer les variables, et d'utiliser des types variés. Mais pour débuter la programmation, il est important de se forcer à déclarer explicitement les variables et les types utilisés.

2) Entrées-sorties

On travaille maintenant sur la notion d'entrée-sortie- Fonctions MsgBox et InputBox.

```
Sub affichage1()
Dim a As Integer
Dim b As Byte
Dim c As String
Dim d As Double
a = -5
MsgBox (a)
```

Pour afficher une valeur, on la passe en paramètre de MsgBox. Elle s'affiche alors dans une petite boite de dialogue. Ce que fait la machine, c'est de convertir la valeur en chaîne de caractères, puis de la visualiser. Ici, on visualise 5.

```
b = 68
MsgBox (b)
c = "quelle valeur?"
MsgBox (c)
d = b / 2
MsgBox (d)
MsgBox ("a vaut :" & a & "  b vaut :" & b & "  c vaut:" & c & "  d vaut :"
& d)
```

On voit ici un exemple d'affichage d'une expression. La machine commence par évaluer l'expression. On rappelle que & est l'opérateur de concaténation de chaînes de caractères. L'expression est calculée grâce aux conversions automatiques de type de nombre vers des chaînes de caractères. Le message affiché dans la boite est :

```
"a vaut :-5 b vaut :68  c vaut:quelle valeur? d vaut :34"
End Sub
```

La fonction InputBox permet d'afficher une boite de dialogue à l'utilisateur, comportant le message passé en paramètre de la fonction. Cette boite comporte une zone blanche, que l'utilisateur de la feuille Excel va renseigner, selon le message que le programme lui aura affiché. La valeur que l'utilisateur indique est le résultat de la fonction. On dit que la fonction renvoie cette valeur. Dans vos débuts de programmeur vous allez toujours procéder ainsi pour interagir avec l'utilisateur de votre macro : demander à cet utilisateur de rentrer une donnée, la ranger en mémoire dans une variable, faire un calcul à partir de cette donnée puis afficher le résultat au moyen de la fonction MsgBox.

```
Sub entree()
Dim a As Integer
a = InputBox("Entrez un entier")
```

Ici, le message Entrez un entier apparaît dans la boite, et l'utilisateur est invité à taper un nombre. Ce nombre est ensuite rangé dans la variable a. Si l'utilisateur tape autre chose qu'un nombre, il y a une erreur d'exécution liée au type de la variable a.

```
MsgBox (a & " et " & a & " font " & a + a)
End Sub
```