EXCEL

Chapitre 4 : Le langage Visual Basic Application

Claude Duvallet

Université du Havre
UFR des Sciences et Techniques
25 rue Philippe Lebon
BP 540
76058 Le Havre Cedex
Courriel : Claude.Duvallet@gmail.com

Généralités sur VBA (1/2)

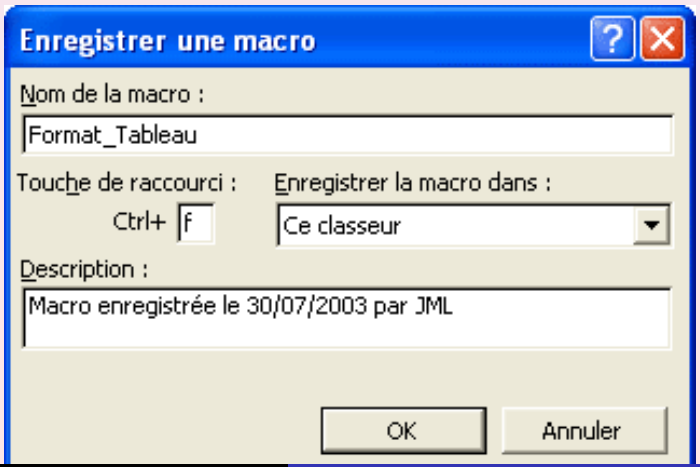
- **VBA** = Visual Basic pour Applications.
- VBA est le langage de programmation des applications de Microsoft Office.
- VBA permet d'automatiser les tâches, de créer des applications complètes, de sécuriser vos saisies et vos documents, de créer de nouveaux menus et de nouvelles fonctions.
- VBA utilise le même langage que Microsoft Visual Basic.
  - VB est un ensemble complet qui permet de développer des applications indépendantes et librement distribuables
  - une application VBA est complètement liée au logiciel sous lequel elle a été créée (une application VBA créée sous Excel ne pourra pas se lancer sur un poste si Excel n'est pas installé).

Généralités sur VBA (2/2)

- Avant Excel utilisait son propre langage de programmation et une application était appelée "macro". Ce terme est resté, mais une macro Excel réalisée en VBA n'est rien d'autre qu'une procédure telle qu'elles sont réalisées sous VB.
- VBA manipule des objets Excel tels que les objets Workbook (classeur), Worskheet (Feuille de calcul), Range(plage de cellule), etc.
- VBA, langage puissant, souple et facile à utiliser permet de réaliser très rapidement des applications.

L'enregistreur de macro-commandes (1/3)

- Excel, comme Word ou PowerPoint possède un outil : l'enregistreur de macros.
- Il crée une macro et transforme en langage VBA toutes les commandes effectuées par l'utilisateur dans l'application hôte.
- Il permet d'automatiser sans aucune connaissance de la programmation certaines de vos tâches et également de se familiariser avec le langage VBA.
- Pour lancer l'enregistreur de macro, il vous suffit dans le menu **Outils** de sélectionner l'option **Macro** puis **Nouvelle macro**.



L'enregistreur de macro-commandes (2/3)

Nom de la macro : Il est possible de renommer la macro. Le nom de la macro doit commencer par une lettre et ne doit pas contenir d'espaces. Utilisez le caractère de soulignement pour séparer les mots.

Touche de raccourci : Il est possible de créer un raccourci clavier pour lancer la macro en saisissant une lettre (minuscule ou majuscule).

Enregistrer la macro dans :

- Ce classeur : la macro ne pourra ensuite s'exécuter que si le classeur dans lequel la macro a été enregistrée est ouvert.
- Le "classeur de macros personnelles" : un nouveau classeur est créé et enregistré dans le dossier "xlstart" de façon que vos macros soient disponibles à chaque fois que vous utilisez Excel.

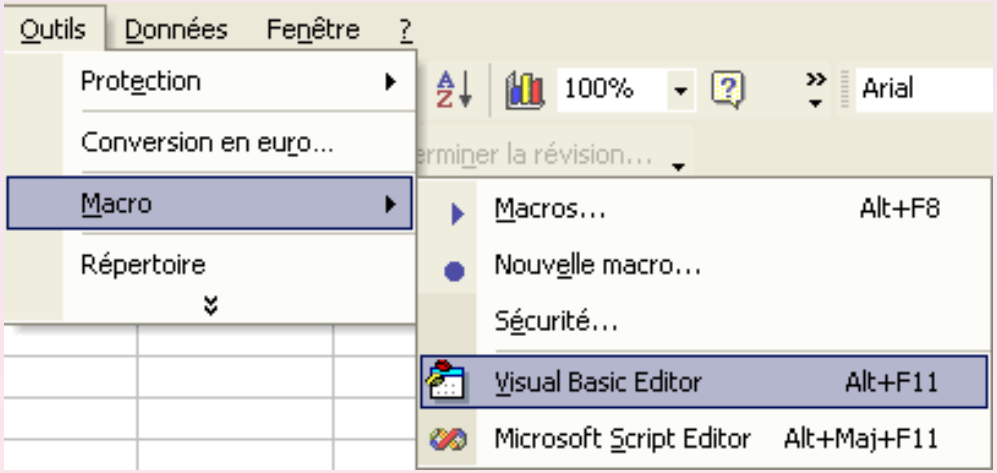
Description : Vous pouvez donner une description à votre macro.

L'enregistreur de macro-commandes (3/3)

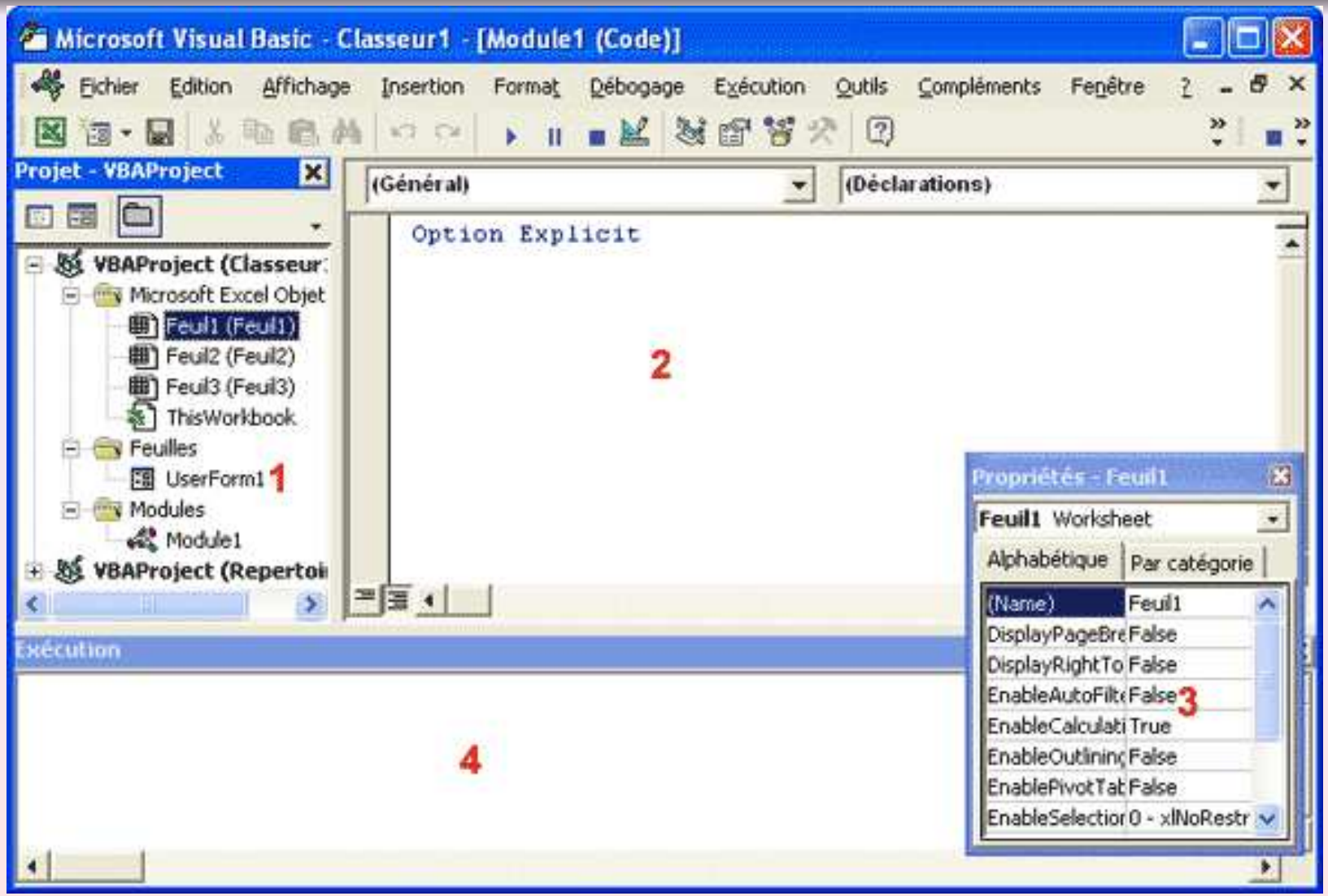
- Vous pouvez ensuite démarrer et arrêter l'enregistrement de la macro au moyen des boutons de la barre d'outils "Macro".
- Toutes les actions effectuées pendant la période d'enregistrement seront inscrites dans la macro en langage VBA même les erreurs ou les annulations.
- Vous pouvez ensuite éditer le code de la macro pour l'examiner ou encore le modifier.

L'éditeur de macro-commandes (1/3)

- L'éditeur de macro, ou VBE (Visual Basic Editor) est l'environnement de programmation de VBA.
- Il se lance par le menu **Outils** puis l'option **Macro** et ensuite l'option **Visual Basic Editor** ou par le raccourci clavier **Alt+F11**.



L'éditeur de macro-commandes (2/3)



L'éditeur de macro-commandes (3/3)

1. **Fenêtre VBAProject** Elle présente les différents projets ouverts et permet de naviguer facilement entre vos différentes feuilles de codes VBA.
2. **Fenêtre Code** C'est l'endroit ou vous allez saisir votre code VBA.
3. **Fenêtre Propriétés** Propriétés de l'objet sélectionné.
4. **Fenêtre Exécution** Elle permet de tester une partie du code. Elle peut s'avérer très utile pour voir comment s'exécutent certaines lignes de code.

Écriture de code VBA (1/3)

- Le code VBA s'écrit dans les modules à l'intérieur de procédures ou de fonctions.
- Dans VBE, il faut commencer par créer un nouveau module par le menu **Insertion** et l'option **Module**. Renommer le module à l'aide de la fenêtre propriétés, la recherche de vos procédures sera plus rapide.
- Une procédure est une suite d'instructions effectuant des actions. Elle commence par `Sub + NomDeLaProcédure` et se termine par `End Sub`. Le nom des procédures doit commencer par une lettre et ne doit pas contenir d'espaces. Utilisez le caractère de soulignement pour séparer les mots.
- Pour déclarer une procédure, taper `Sub` et son nom puis taper `Entrée`. VBE ajoute automatiquement les parenthèses et la ligne `End Sub`.

Écriture de code VBA (2/3)

- Une fonction est une procédure qui renvoie une valeur. Elle se déclare de la même façon qu'une procédure.
- En général, on écrit une instruction par ligne.
- Il est possible d'ajouter des lignes de commentaire entre les lignes d'instruction ou au bout de celles-ci. Les commentaires sont précédés d'une apostrophe et prennent une couleur différente.
- Il n'y a pas de limite de caractères pour chaque ligne d'instruction. Il est toutefois possible d'écrire une instruction sur plusieurs lignes afin d'augmenter la visibilité du code. Pour cela, il faut ajouter le caractère de soulignement avant le passage à la ligne.
- L'option **Info express automatique** permet d'afficher les informations relatives à la fonction que vous venez de taper.

Écriture de code VBA (3/3)

- Il est également possible d'obtenir de l'aide à tout moment par la combinaison de touches **Crtl+j**.
- La vérification automatique de la syntaxe vous alerte si il y a une erreur dans l'écriture du code et la ligne de code change de couleur.
- Chaque procédure `Sub` ou `Function` peut être appelée de n'importe qu'elle autre procédure du projet. Pour restreindre la portée d'une procédure au module, déclarez-la **Private**.
- À l'intérieur de vos procédures, écrivez vos instructions en minuscules, VBE se chargera de transformer votre code par des majuscules.

Les objets

VBA manipule les objets de l'application hôte. Chaque objet possède des propriétés et des méthodes.

- Chaque objet représente un élément de l'application : un classeur, une feuille de calcul, une cellule, un bouton, etc.
- Par exemple, Excel représente l'objet **Application**, **Workbook** l'objet classeur, **Worksheet** l'objet feuille de calcul, etc.
- Tous les objets de même type forment une collection comme, par exemple, toutes les feuilles de calcul d'un classeur. Chaque élément est alors identifié par son nom ou par un index.
- Par exemple, pour faire référence à la Feuil2, on va utiliser `Worksheets(2)` ou `Worksheets("Feuil2")`
- Chaque objet peut avoir ses propres objets. Un classeur possède des feuilles qui possèdent des cellules. Pour faire référence à une cellule, on peut donc utiliser :

```
Application.Workbooks(1).Worksheets("Feuil2").Range("A1")
```

Les propriétés

- Une propriété correspond à une particularité de l'objet.
- La valeur d'une cellule, sa couleur, sa taille, etc. sont des propriétés de l'objet `Range`.
- Les objets sont séparés de leurs propriétés par un point. On écrira ainsi Cellule.Propriété=valeur.

```
'Mettre la valeur 10 dans la cellule A1
Range("A1").Value = 10
```

- Une propriété peut également faire référence à un état de l'objet. Par exemple, si on veut masquer la feuille de calcul "Feuil2", on écrira :

```
Worksheets("Feuil2").Visible = False
```

Les méthodes

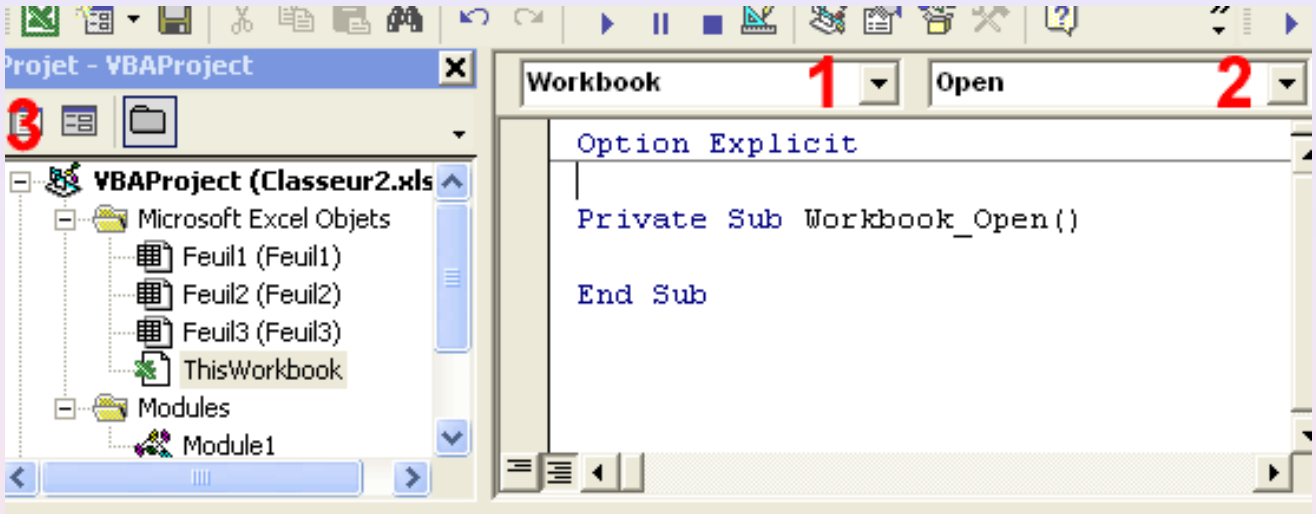
- On peut considérer qu'une méthode est une opération que réalise un objet.
- Les méthodes peuvent être considérées comme des verbes tels que ouvrir, fermer, sélectionner, enregistrer, imprimer, effacer, etc.
- Les objets sont séparés de leurs méthodes par un point. Par exemple, pour sélectionner la feuille de calcul nommé "Feuil2", on écrira : `Worksheets("Feuil2").Select`
- Lorsque l'on fait appel à plusieurs propriétés ou méthodes d'un même objet, on fera appel au bloc d'instruction **With Objet Instructions End With**. Cette instruction rend le code souvent plus facile à lire et plus rapide a exécuter.

```
'Mettre la valeur 10 dans la cellule A1, la police en gras
'et en italique et copier la cellule.
With Worksheets("Feuil2").Range("A1")
    .Value = 10
    .Font.Bold = True
    .Font.Italic = True
    .Copy
End With
```

Les événements (1/3)

- Pour qu'une macro se déclenche, il faut qu'un événement (un clic sur un bouton, l'ouverture d'un classeur, etc.) se produise. Sans événements, rien ne peut se produire.
- Il existe deux types d'événements : les évènements liés aux objets et les événements liés aux objets.
- Les principaux objets pouvant déclencher une macro sont :
  - Un classeur
  - Une feuille de travail
  - Une boite de dialogue
- Chacun de ces objets possède leur propre module. Pour y accéder, il faut lancer l'éditeur de macro.

Les événements (2/3)



- Pour créer une procédure événementielle liée à un classeur, sélectionner le classeur **ThisWorkbook** puis cliquez sur l'icône **3** (ou plus simplement double-clic sur **ThisWorkbook**).
- Vous accédez ainsi au module lié à l'objet. Sélectionnez **Workbook** dans la liste **1** puis sur l'événement désiré dans la liste **2**.

Les événements (3/3)

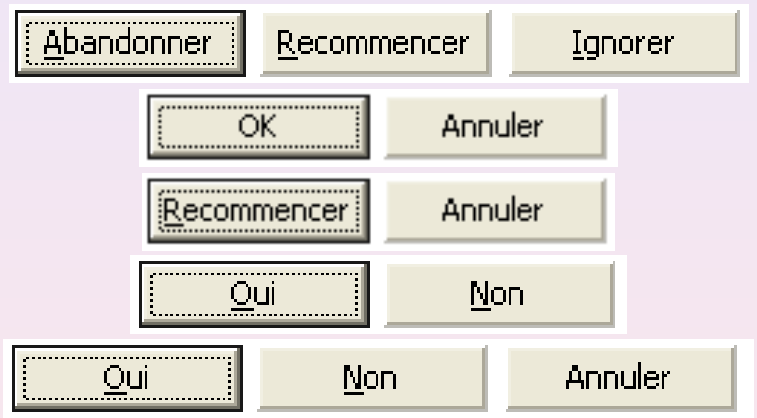
- La création d'une procédure événementielle liée à une feuille de calcul se fait de la même façon.
- Une macro peut également être déclenchée à une heure donnée (**OnTime**) ou lorsque l'utilisateur appuie sur une touche (**OnKey**).
- Le déclenchement d'une macro nommée "Test" à 15 Heures se fait par la ligne d'instruction suivante : `Application.OnTime TimeValue("15:00:00"), "Test"`.
- Le déclenchement d'une macro nommée "Test" lorsque l'utilisateur appuie sur la touche "F1" se fait par la ligne d'instruction suivante : `Application.OnKey "F1", "Test"`.

Les messages (1/4)

- Lors d'une procédure, les messages servent à communiquer avec l'utilisateur.
- Il existe des messages qui donnent de l'information et d'autres qui en demandent.
- Les **MsgBox** peuvent simplement donner une information. La procédure est alors stoppée tant que l'utilisateur n'a pas cliqué sur le bouton.
- Le texte peut-être affiché sur plusieurs lignes en utilisant le code retour chariot `chr(13)` ou le code retour ligne `chr(10)`.
- Vous pouvez ajouter une icône concernant le type de message à afficher.

Les messages (2/4)

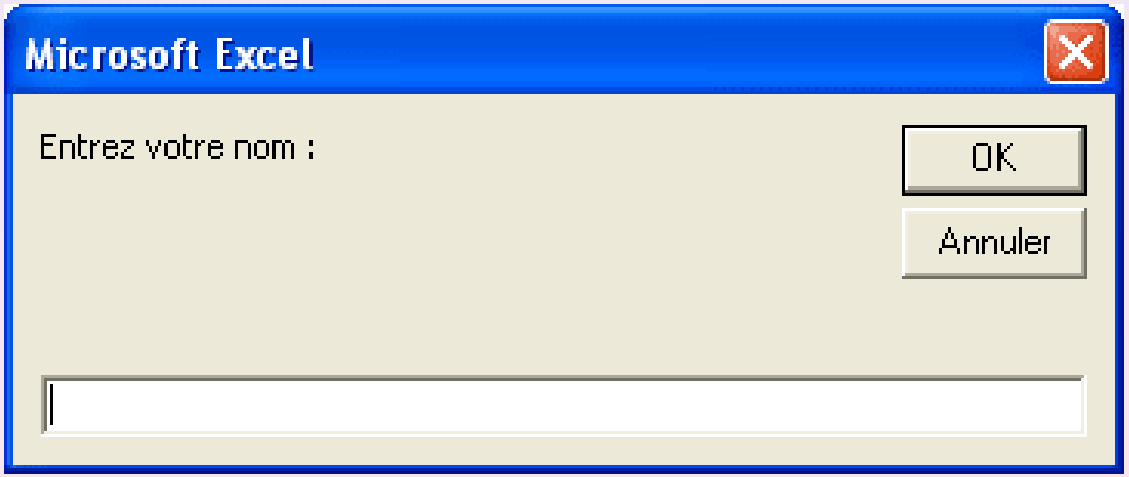
- Les MsgBox peuvent également demander une information à l'utilisateur. Dans ce cas, la boîte de message comprend plusieurs boutons.



- La syntaxe est `MsgBox ("Message", attribut bouton ).`
- Vous pouvez également y ajouter les icônes et personnaliser le titre de la fenêtre en utilisant la syntaxe : `Msgbox ("Message", attribut bouton + attribut icône, "titre de la fenêtre").`

Les messages (3/4)

- Les **InputBox** sont des boîtes de dialogue dans lesquelles l'utilisateur est invité à entrer des données. La syntaxe est : `InputBox ("Message").`



- Comme pour les **MsgBox**, vous pouvez changer le titre de la fenêtre.

Les messages (4/4)

- Vous pouvez également entrer une valeur par défaut dans la zone de saisie. La syntaxe devient : `InputBox ("Message", "Titre de la fenêtre", "Valeur par défaut")`.
- La valeur saisie peut être récupérée dans une variable. Si l'utilisateur clique sur le bouton "Annuler", la variable renvoie une chaîne de longueur nulle ("").
- Vous pouvez également écrire un message dans la barre d'état de l'application. La syntaxe est : `Application.StatusBar = "Message"`.
- À la fin de la procédure, pensez à supprimer le message de la barre d'état par la ligne d'instruction : `Application.StatusBar = False`.

Les variables (1/3)

- Lors d'une procédure, les variables servent à stocker toutes sortes de données (des valeurs numériques, du texte, des valeurs logiques, des dates, ...).
- Elles peuvent également faire référence à un objet.
- Suivant les données que la variable recevra, on lui affectera un type différent.
- Pour rendre obligatoire la déclaration de variables, placez l'instruction `Option Explicit` sur la première ligne du module ou cochez l'option **Déclaration des variables obligatoires** dans le menu **Outils** sous-menu **Options** de l'éditeur de macros.
- La déclaration explicite d'une variable se fait par le mot Dim (abréviation de Dimension).

Les variables (2/3)

- Le nombre maximum de caractères du nom de la variable est de 255. Il ne doit pas commencer par un chiffre et ne doit pas contenir d'espaces. La syntaxe est `Dim NomDeLaVariable as Type`.
- La portée d'une variable est différente suivant l'endroit et la façon dont elle est déclarée.
- Une variable déclarée à l'intérieur d'une procédure est dite "Locale". Elle peut-être déclarer par les mots `Dim`, `Static` ou `Private`.
- Dès que la procédure est terminée, la variable n'est plus chargée en mémoire sauf si elle est déclarée par le mot `Static`.
- Une variable locale est généralement placée juste après la déclaration de la procédure.

Les variables (3/3)

- Une variable peut être "locale au module" si celle-ci est déclarée avant la première procédure d'un module. Toutes les procédures du module peuvent alors lui faire appel. Elle est déclarée par les mots `Dim` ou `Private`.
- Une variable peut également être accessible à tous les modules d'un projet. On dit alors qu'elle est publique. Elle est déclarée par le mot `Public`. Elle ne peut pas être déclarée dans un module de **Feuille** ou dans un module de **UserForm**.
- Une variable peut garder toujours la même valeur lors de l'exécution d'un programme. Dans ce cas, elle est déclarée par les mots **Const** ou **Public Const**.

Les classeurs et les feuilles de calcul

- Les **classeurs** sont désignés par le mot `Workbook`. Ils peuvent être ouverts, fermés, enregistrés, activés, masqués, supprimés,... par une instruction VB.
- Les **feuilles de calcul** sont désignées par le mot `Worksheet`. Comme les `Workbook`, ces objets possèdent de nombreuses propriétés et méthodes.

Les cellules

- Une plage de cellules est désignée par l'objet `Range`. Pour faire référence à la plage de cellule "A1 :B10", on utilisera `Range("A1:B10")`.
- L'objet `Range` permet également de faire référence à plusieurs plages de cellules non contiguës.
- Pour faire référence à une seule cellule, on utilisera l'objet `Range("Référence de la cellule)` ou `Cells(Numéro de ligne, Numéro de colonne)`.
- VB vous permet également de changer le format des cellules (polices, couleur, encadrement ...).
- À partir d'une cellule de référence, vous pouvez faire appel aux autres cellules par l'instruction `Offset`. La syntaxe est `Range(Cellule de référence).Offset(Nombre de lignes, Nombre de colonne)`.
- Les arguments (Nombre de lignes, Nombre de colonnes) de l'instruction `Offset` sont facultatifs et leur valeur par défaut est 0.

Les instructions conditionnelles

- Elles peuvent déterminer la valeur que prennent les variables, arrêter une procédure, appeler une procédure, quitter une boucle, atteindre une étiquette.
- L'instruction la plus courante dans VB est la condition **If** `condition` **Then** `instruction_vrai`.
- Si la valeur vraie possède plusieurs lignes d'instructions, la syntaxe devient **If** `condition` **Then** `instructions_vrai` **End If**.
- Une autre possibilité de syntaxe est la suivante : **If** `condition` **Then** `instructions_si_vrai` **Else** `instructions_si_faux` **End If**.
- Dans le cas de conditions multiples, il est préférable d'utiliser le bloc d'instruction **Select Case** `expression` **Case** `valeur instructions` **Case Else** `instructions` **End Select**.

Les boucles (1/2)

- Les boucles le plus souvent utilisés sont les boucles **For ... Next**.
- Elles permettent de répéter un nombre de fois défini un bloc d'instructions.
- Elles utilisent une variable qui est incrémentée ou décrémentée à chaque répétition.
- La variable peut être incrémentée d'une valeur différente de 1 par le mot **Step**.
- La variable peut également être décrémentée, le mot **Step** est alors obligatoire.

Les boucles (2/2)

- À l'intérieur d'un bloc d'instruction **For Next**, l'instruction **Exit For**, peut quitter la boucle avant que la variable n'est atteint sa dernière valeur.
- Pour répéter un bloc d'instructions pour chaque objet d'une collection ou pour chaque élément d'un tableau, on utilisera le bloc d'instruction **For Each** *Objet* **In Collection Next**.
- On peut également utiliser l'instruction **Exit For** pour sortir d'un bloc d'instruction **For Each ... Next**.

Les boucles conditionnelles

- Les boucles **While** *condition* `instructions` **Wend** exécutent un bloc d'instruction tant que la condition est vraie.
- Les boucles **Do Loop** sont mieux structurées que les boucles **While Wend**. On peut à tout moment sortir d'une boucle **Do Loop** par l'instruction **Exit Do**.
- La boucle **Do While** *condition* instructions **Loop** exécute un bloc d'instruction tout pendant que la condition est vraie.
- Dans la boucle **Do** `instructions` **Loop While** *condition*, le bloc d'instruction est exécuté une fois avant que la condition soit testée.
- Pour sortir d'une boucle, on utilise l'instruction **Exit Do**.
- Les boucles **Do Until** sont identiques aux boucles **Do While**, seulement le bloc d'instruction est répété tout pendant que la condition n'est pas vraie.

Les fonctions de texte (1/3)

- VBA possède des fonctions permettant d'extraire une chaîne de caractères d'un texte.
- La fonction **Len** renvoie le nombre de caractères d'un texte.
- La fonction **Left** renvoie un nombre de caractères en partant de la gauche. La syntaxe est **Left**(Texte, Nombre de caractères).
- La fonction **Right** renvoie un nombre de caractères en partant de la droite. La syntaxe est **Right**(Texte, Nombre de caractères).
- La fonction **Mid** renvoie un nombre de caractères en partant d'un caractère défini. La syntaxe est **Mid**(Texte, Départ, Nombre de caractères). Si le Nombre de caractères n'est pas indiqué, la fonction renvoie tous les caractères à partir de la position départ.
- La fonction **LTrim** supprime les espaces se trouvant avant la chaîne de caractères.

Les fonctions de texte (2/3)

- La fonction **RTrim** supprime les espaces se trouvant après la chaîne de caractères.
- La fonction **Trim** supprime les espaces se trouvant avant et après la chaîne de caractères.
- La fonction **Ucase** convertie le texte en majuscules.
- La fonction **Lcase** convertie le texte en minuscules.
- La fonction **Application.Proper** convertie le texte en nom propre.
- La fonction **Replace** permet de remplacer une chaîne de caractères par une autre. Cette fonction possède des arguments facultatifs permettant de déterminer la position du premier remplacement et le nombre de remplacement à effectuer. La syntaxe est **Replace**(Texte, Chaîne à remplacer, chaîne de remplacement, Départ, Nombre de remplacement).

Les fonctions de texte (3/3)

- La fonction **Val** renvoie la valeur numérique d'une chaîne de caractères. Si la chaîne de caractères est composée de chiffres et de lettres, la valeur s'arrête au premier caractère non numérique.
- La fonction **IsNumeric** permet de tester si une chaîne de caractères est numérique et renvoie une valeur de type **Boolean**.
- La fonction **IsDate** permet de tester si une chaîne de caractères est une date. Elle renvoie une valeur de type **Boolean**.
- Certaines fonctions permettent de convertir des données en d'un type défini. Par exemple, **CDate** va convertir des données en date.
- Le format des dates et heures est défini par la fonction **Format**. La syntaxe est **Format**(MaDate, Format). La fonction **Format** permet également de formater les nombres.

Remerciements

Cette présentation est appelée à être enrichie et améliorée.
Cette présentation a été réalisée au moyen de divers supports dont certains trouvés sur Internet et notamment ceux qui suivent :
`http://www.lecompagnon.info/excel/index.html`
`http://perso.wanadoo.fr/jml85/Pages/cours_VBA.htm`