EXCEL 2010

Visual Basic

Table des matières

Introduction

EXCEL VBA (Visual Basic pour Application) est un langage de programmation permettant d'utiliser du code VisualBasic pour exécuter les nombreuses fonctionnalités d'Excel

Un programme écrit en VBA est souvent appelé une macro.

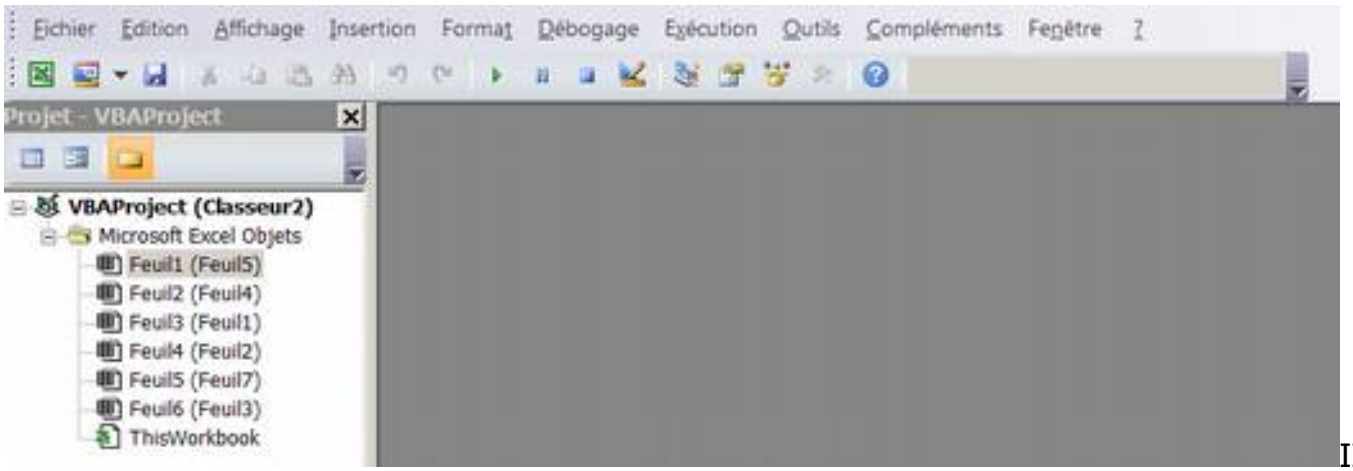
Les macros permettent notamment d'automatiser des tâches répétitives réalisées sous EXCEL. Elles peuvent aussi être utilisées pour créer des boîtes de dialogue afin de rendre une application développée sous EXCEL plus conviviale.

Une macro peut être créée en utilisant l'enregistreur de macros, qui ne nécessite aucune connaissance du langage VBA.

Cependant une macro ainsi créée ne s'exécutera que sur un ensemble de cellules données et le code produit ne sera pas toujours très efficace. Pour pouvoir créer des macros propres à ses besoins, efficaces et interactives, il faut apprendre à programmer en VBA.

L'environnement Visual Basic Editor

Ouvrez l'éditeur en tapant ALT – F11
Vous obtenez :



II

INSERTION – Module
Vous obtenez :

C'est dans cette fenêtre que vous saisirez le code Visual Basic.

Premiers pas

Procédures

Une macro est appelée en VBA une **procédure**, c'est une suite d'instructions qui ne retourne pas de valeur.
VBA permet également d'écrire des fonctions. Une fonction est une suite d'instructions qui retourne une valeur. Elles seront étudiées plus loin dans ce cours.

L'enregistreur de macro ne génère que des procédures. Une procédure commence par le mot clé Sub suivi du nom de la procédure et d'une liste d'arguments entre parenthèses, qui peut être vide. Elle se termine par le mot clé End Sub.

Entre le Sub et le End Sub se trouvent la suite des **instructions** qui seront exécutées séquentiellement.

Une procédure a la syntaxe suivante :
Sub NomProcédure([argument_1,..., argument_n])
Instructions
...
End Sub
Remarque : Les crochets [ ] signifient que les arguments sont facultatifs.

Instructions

Une instruction exécute une tâche précise. Elle est généralement écrite sur une seule ligne.
Exemple :
Range("A1").Select

Exécuter une macro

1. Activez l'onglet « Développeur » (FICHIER – Options – Personnaliser le ruban)
2. Dans le groupe « Code » de l'onglet « Développeur » cliquez sur le bouton « Macros » :

3. Sélectionnez votre macro dans la liste
4. Cliquez sur le bouton « Exécuter » :

Modèle Objet

Les Objets

VBA est un langage de programmation orientée objet attaché à une application. Tout est objet.
Exemple d'objets :

- EXCEL est un objet Application ;
- Un classeur est un objet Workbook ;
- Une feuille de calcul est un objet Worksheet ;
- Une plage de cellules (qui peut se limiter à une cellule) est un objet Range.

Les Collections

De nombreux objets appartiennent à une collection d'objets, la collection étant elle-même un objet.
Exemple de collections :
Dans l'objetApplication, il existe une collection Workbooks qui contient tous les objets Workbook ouverts. Chaque objet Workbook comporte une collection Worksheets qui contient tous les objets Worksheet de ce classeur. Chaque objet Worksheet contient des objets Range.

Accès aux Objets

VBA permet de faire référence à un objet de différentes façons. Nous illustrons l'accès aux objets EXCEL à travers plusieurs exemples.
Pour faire référence à une plage de cellules donnée, il faut utiliser l'objet Range.
Exemple : Range("A1:B4") permet de faire référence à la plage de cellules A1:B4 et renvoie un objet Range.
Pour faire référence à un objet dans une collection, on peut soit utiliser le numéro de sa position dans la collection, soit son nom.
Exemple :
Worksheets(1) permet de faire référence à la première feuille de calcul du classeur actif et renvoie un objet Worksheet. Workbooks("Classeur1.xls").Worksheets("Feuil1") permet de faire référence à la feuille de calcul de nom Feuil1 du classeur de nom Classeur1.xls et renvoie un objet Worksheet.

Propriétés des objets

Chaque objet est défini par un ensemble de propriétés qui représentent les caractéristiques de l'objet.
Pour faire référence à une propriété d'un objet donné, il faut utiliser la syntaxe Objet.Propriété.

Exemple :

Propriété	Value
Utilisation	Désigne la valeur de l'objet spécifié
Exemple	Range("A1").Value = "Atem Formation"
Résultat	Affecte la valeur Atem Formation à la cellule A1

Méthodes

Les méthodes représentent les actions qui peuvent être effectuées par un objet ou que l'on peut appliquer à un objet.
Pour faire référence à une méthode d'un objet donné, il faut utiliser la syntaxe
Objet.Méthode.
Exemple :

Méthode	Select
Utilisation	Sélectionne une cellule ou une plage de cellules
Exemple	Range("A1").Select
Résultat	Sélectionne la cellule A1

Evénements

Un événement est une action reconnue par un objet. Cette reconnaissance permet de déclencher l'exécution d'une procédure lorsque cet événement survient.
Exemples d'événements :

- Un clic souris
- La frappe d'une touche au clavier

Pour qu'un objet réponde à un événement, il faut écrire du code VBA dans la procédure associée à l'événement considéré.

Exemple

Ecrire une procédure qui colorie chaque cellule (ou plage de cellules) sélectionnée(s) de la Feuille 2 en vert.
Méthode :
L'événement SelectionChange devant se produire lors de la sélection il faudra saisir le code dans le module de code de la Feuille 2. Pour cela, depuis l'éditeur, faites un double-clic sur Feuil2 :
Sélectionnez la procédure « SelectionChange » dans l'objet « Worksheet » :

Saisissez le code suivant :

```
Private Sub Worksheet_SelectionChange(ByVal Target As Range)
 Target.Interior.Color = 47896
End Sub
```

La Notion de Variable

Introduction

Une variable permet de stocker une valeur pouvant être modifiée au cours de l'exécution d'un programme.

Le mot clé Dim permet de déclarer explicitement une variable :
Dim NomVariable
où NomVariable est une suite de caractères formés avec des lettres, des chiffres et le caractère souligné _. Le premier caractère est obligatoirement une lettre. Les minuscules sont distinguées des majuscules.

Le langage VBA permet de ne pas déclarer explicitement les variables. Un des intérêts de la déclaration explicite des variables est d'éviter les erreurs de frappes malheureusement très fréquentes en programmation.

Il est donc préférable de forcer la déclaration explicite des variables en VBA. Pour ce faire, il suffit de placer l'instruction Option Explicit en haut des modules de code avant toutes procédures et toutes fonctions.

Différents types de variables

La plupart des langages de programmation imposent de déterminer le type de données qui peut être stockée dans une variable lors de sa déclaration. En VBA, ce n'est pas obligatoire.

Par défaut, une variable non typée est du type Variant, qui permet de stocker n'importe quel type de données.

Dans un soucis d'efficacité du code, il est préférable de typer ses variables. Une variable de type Variant prend, en effet, plus de mémoire que n'importe quel autre type et utilise plus de ressource système pour son traitement (identification du type effectif de la variable et éventuellement conversion).

Déclaration des variables

La déclaration du type d'une variable se fait comme suit : Dim NomVariable As Type où Type détermine le type de la valeur qui peut être stockée dans la variable.

Types de variables

Byte, Integer et Long	pour les entiers
Single, Double et Currency	pour les réels
Boolean	pour les booléens (True ou False)
String	pour les chaînes de caractères
Date	pour les dates
Object	pour faire référence à un objet
Variant	our n'importe quel type

Exemple :
Dim Nom As String

Nom = "Jean "

Dim Age As Byte
Age = 23

Portée d'une variable

La portée d'une variable définit quelles procédures ou fonctions peuvent utiliser cette variable.

Les variables déclarées à l'intérieur d'une procédure ou d'une fonction ne sont accessibles qu'à l'intérieur de cette procédure ou de cette fonction.

Les variables déclarées au début du module à l'extérieur de toute procédure et de toute fonction sont accessibles à l'ensemble des procédures et des fonctions de ce module

Les variables déclarées au début d'un module à l'extérieur de toute procédure et de toute fonction à l'aide du mot clé Public sont accessibles à l'ensemble des procédures et des fonctions d'un projet.

Exemple :
Public Age As Integer

La déclaration des constantes

Une constante permet d'attribuer un nom à une valeur fixe. La déclaration d'une constante se fait à l'aide du mot clé Const :
Const NomConstante [As Type] = valeur

Une constante a la même portée qu'une variable.

Entrées/Sorties

Fonction MsgBox

La fonction MsgBox affiche un message dans une boîte de dialogue, attend que l'utilisateur clique sur un bouton, puis renvoie un entier indiquant le bouton choisi par l'utilisateur. Elle a la syntaxe suivante (les arguments entre crochets sont facultatifs) :

MsgBox(prompt[,buttons][,title][,helpfile, context])

prompt	message affiché dans la boîte de dialogue
buttons	correspond au type de boutons de la boîte de dialogue
title	titre de la boîte de dialogue

Exemple :
reponse = MsgBox("Avez-vous dîné ?", vbYesNo, "Repas") affiche :

La réponse de l'utilisateur est stockée dans la variable reponse.

Fonction InputBox

La fonction InputBox affiche une invite dans une boîte de dialogue, attend que l'utilisateur tape du texte ou clique sur un bouton, puis renvoie le contenu de la zone de texte sous la forme d'une chaîne de caractère. Elle a la syntaxe suivante (les arguments entre crochets sont facultatifs) :

InputBox(prompt[,title][,default][,xpos][,ypos] [,helpfile,context])

prompt	message affiché dans la boîte de dialogue
title	titre de la boîte de dialogue

Exemple :
reponse = InputBox("Quel âge avez-vous ?", "Age") affiche :

La réponse de l'utilisateur est stockée dans la variable reponse.

Tests Simples

Instruction: If … Then … Else

VBA permet de traiter les énoncés conditionnels à l'aide de l'instruction If. On distingue l'instruction If écrite sur une seule ligne et le bloc If.

L'instruction If écrite sur une seule ligne

L'instruction If écrite sur une seule ligne a la syntaxe suivante :
If condition Then Instruction_si_vrai [Else Instruction_si_faux]

Exemple :
If age>=60 Then Call MsgBox("Vétéran") Else Call MsgBox("Autre")

Le bloc If

La structure de bloc If permet aux sections Then et Else de contenir plusieurs instructions. Elle peut avoir l'une des deux syntaxes suivantes :

```
If condition Then
    Instruction 1_si_vrai
    Instruction 2_si_vrai
    ....
End If
```

Instructions de Boucle

VBA fournit deux structures pour répéter en boucle les mêmes instructions :
l'instruction Do...Loop et l'instruction For...Next.

 Une boucle peut parfois tourner indéfiniment ! Pour la stopper tapez CTRL Pause

Boucle Do...Loop

La boucle Do...Loop est utilisée lorsque l'on veut que la répétition des instructions s'arrête quand une condition donnée est vérifiée. On distingue plusieurs syntaxes possibles.

Do While condition Instructions Loop	Les instructions sont exécutées tant que la condition est vraie
Do Until condition Instructions Loop	Les instructions sont exécutées jusqu'à ce que la condition devienne vraie

Boucle For...Next

La boucle For...Next est utilisée lorsque l'on connaît à l'avance le nombre de fois où l'on veut que les instructions soient répétées.

Syntaxe :

```
For compteur = nbdébut To nbfin [Step nbpas]
 Instructions
Next compteur
```

Les instructions de la boucle For...Next sont exécutées un nombre déterminé de fois. Lorsque l'option Step n'est pas renseignée, le compteur est incrémenté de 1 à chaque itération et les instructions sont exécutées nbdébut - nbfin fois.

Exemple :

La procédure suivante calcule la somme des n premiers entiers :

```
Sub SommeEntiers()

 n = InputBox("Donner un entier :", "Saisie entier")
 For compteur = 1 To n
 somme = somme + compteur
 Next compteur
 Call MsgBox("Somme des " & n & " premiers entiers : " & somme)
End Sub
```

Boucle For Each... Next

La boucle For Each...Next permet d'accéder aux objets d'une collection.
Exemple :

```
Sub ColorieCellule()
 Dim MaPlage As Range
 Dim Cellule As Range
 Dim i As Integer
 Set MaPlage = Range("A1:B5")
 For Each Cellule In MaPlage
 Cellule.Interior.ColorIndex = i
 i = i + 1
 Next Cellule
End Sub
```

Sortir d'une boucle

L'instruction Exit permet de quitter un bloc Do...Loop, For...Next, Function, ou Sub. Sa syntaxe est la suivante :

```
Exit Do
Exit For
Exit Function
Exit Sub
```

Exemple :
La boucle suivante s'arrête lorsque la réponse rep est n ou N :

```
 Do
 Rep = InputBox("Voulez-vous continuer ? (O/N)")
 If LCase(rep) = "n" Then Exit Do
 Loop
```

Les Fonctions

Présentation

VBA offre la possibilité de créer ses propres fonctions, qui peuvent être utilisées dans EXCEL comme n'importe quelle fonction intégrée.

Syntaxe

Une fonction est une suite d'instructions qui retourne une valeur. Elle commence par le mot clé Function suivi du nom de la fonction et d'une liste d'arguments entre parenthèses, qui peut être vide. Elle se termine par le mot clé End Function.Syntaxe
Une fonction a la syntaxe suivante :

```
Function NomFonction([argument_1,..., argument_n])
 Instructions
 ...
 NomFonction = Expression 'valeur de retour
 ...
End Function
```

Une fonction indique la valeur à retourner en initialisant son nom avec la valeur de retour.

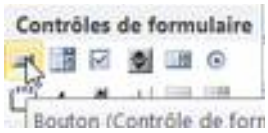
Attention : Dans VBA, les arguments d'une fonction sont séparés par des virgules, alors que dans EXCEL ils sont séparés par des points-virgules.

Déclenchement des macros

Avec un bouton dans la feuille

1. Dans le groupe « Contrôles » de l'onglet « Développeur » cliquez sur le bouton « Insérer » :

2. Cliquez sur le bouton « Bouton » :



3. Avec la souris délimitez un rectangle. Relachez. Vous obtenez :

4. Sélectionnez dans la liste la macro à affecter.
5. Cliquez sur le bouton OK

Vous obtenez :

6. Cliquez à l'intérieur du bouton pour modifier son libellé. Par exemple :

Avec un bouton dans le ruban

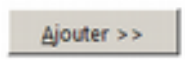
Attention ceci voudra dire que cette macro pourra être exécutée depuis n'importe quel fichier. Elle devra donc être placée dans le classeur de macros personnelles. (et celui-ci devra être affiché)

FICHIER – Options – Personnaliser le ruban

Sélectionnez « Macros » :

Sélectionnez votre macro dans la liste :

Et cliquez sur le bouton « Ajouter » :



Débogage des macros

Le débogage consiste à régler les erreurs directement liées au code d'un programme. Trois types d'erreur peuvent affecter un programme écrit en VBA :

- Des erreurs de compilation qui surviennent lorsque VBA rencontre une instruction qu'il ne reconnaît pas
- Des erreurs d'exécution
- Des erreurs logiques : le programme s'exécute mais le résultat obtenu ne correspond pas à celui attendu.

Erreurs de compilation

Dans l 'Editeur :
DEBOGAGE – Compiler VBAProject

Erreurs d'exécution

Si une erreur d'exécution est générée, l'instruction coupable est mise en évidence. Corriger l'erreur.

Erreurs logiques

Ce sont les plus difficiles à corriger. Vous pouvez placer un point d'arrêt devant une instruction. VBA, lors de l'exécution de la macro, s'arrêtera alors à cet endroit précis du code. Vous pourrez ensuite avancer l'exécution pas à pas et vérifier au fur et à mesure les valeurs des variables.

Placer un point d'arrêt :
Cliquez devant la ligne de l'instruction (dans la zone grise) . Exemple :

```
Sub damier()
    nb = Val(InputBox("Nombre de cases ?"))
   Range(Cells(1, 1), Cells(nb, nb)).Select
    Selection.Interior.Color = 0
    For c = 1 To nb
        decalage = c Mod (2)
        For l = 1 + decalage To nb Step 2
            Cells(l, c).Select
            Selection.Interior.Color = 255
        Next
    Next
End Sub
```

Avancer pas à pas
AFFICHAGE – Barre d'outils – Débogage
Cliquez sur le bouton « Pas à pas détaillé » pour avancer étape par étape :

Espionner une variable
Posez la souris sur la variable. Une infobulle vous donner sa valeur.

Piloter Word

Ressource : https://vb.developpez.com/faqvba/

Rajouter la bibliothèque Word

OUTILS - Références
Cocher « Microsoft Word 14.0 Object Library »

Ouvrir un fichier

```
Set WordObj = CreateObject("Word.Application.8")
Set WordDoc = WordObj.Documents.Add(Template:=strfichier, NewTemplate:=False,
DocumentType:=0)
WordObj.Visible = True
```

strfichier = chemin complet du fichier à ouvrir

Atteindre un signet

```
WordObj.Selection.GoTo what:=wdGoToBookmark, Name:="signet"
```

signet = nom du signet

Ecrire à l'emplacement du curseur

```
WordObj.Selection.TypeText Text:= "texte"
```

Sélectionner la ligne que l'on vient d'écrire

```
WordObj.Selection.HomeKey Unit:=wdLine, Extend:=wdExtend
```

Formater le texte

Cet exemple met le texte (préalablement sélectionné) en gras et centré :

```
With WordObj.Selection.Range
        .Bold = True
        .ParagraphFormat.Alignment = wdAlignParagraphCenter
End With
```

Cet exemple applique le style "Titre 1" au paragraphe sélectionné :

```
WordObj.Selection.Range.Style = "Titre 1"
```

Ajouter un paragraphe à la suite

```
WordObj.ActiveDocument.Paragraphs.Add
```