PROGRAMMATION VBA SOUS MICROSOFT EXCEL

SOMMAIRE

1. Qu'est-ce que VBA ?

VBA est l'acronyme de **V**isual **B**asic for **A**pplications. Il s'agit du langage de programmation de l'ensemble des applications Microsoft Office (Excel, Word, Access, etc.), introduit à partir de la version 97. VBA permet d'automatiser certaines tâches rébarbatives ou répétitives, mais encore de réaliser des applications complètes. Les **procédures** développées sont également appelées des **macros**.

VBA est un langage de programmation orienté objet : on accède ainsi aux différents éléments de l'application (classeur, feuilles de calcul, graphiques, cellules, etc.) en utilisant ce qu'on appelle des objets, qui représentent précisément ces éléments. Par exemple, pour renommer le classeur Classeur1, il suffira d'entrer :

WorkBooks("Classeur1").name = nouveau_nom

Pour chaque objet est défini un ensemble de **propriétés** et de **méthodes**, auxquelles on accède au moyen d'instructions.
Les propriétés sont des caractéristiques modifiables ou non de l'objet.
Par exemple, Name est ici une propriété de l'objet WorkBooks("Classeur1").

Une propriété peut également être un objet. Par exemple, pour changer le nom de la feuille de calcul Feuil1 du classeur Classeur1, on écrira :

Workbooks("Classeur1").Worksheets("Feuil1").Name = nouveau_nom

Les méthodes sont les actions rattachées à l'objet en question. Par exemple, la méthode close de l'objet WorkBooks("Classeur1") sert à fermer le classeur :

WorkBooks("Classeur1").close

L'ensemble des objets contenus dans un objet particulier forme ce que l'on appelle une collection.

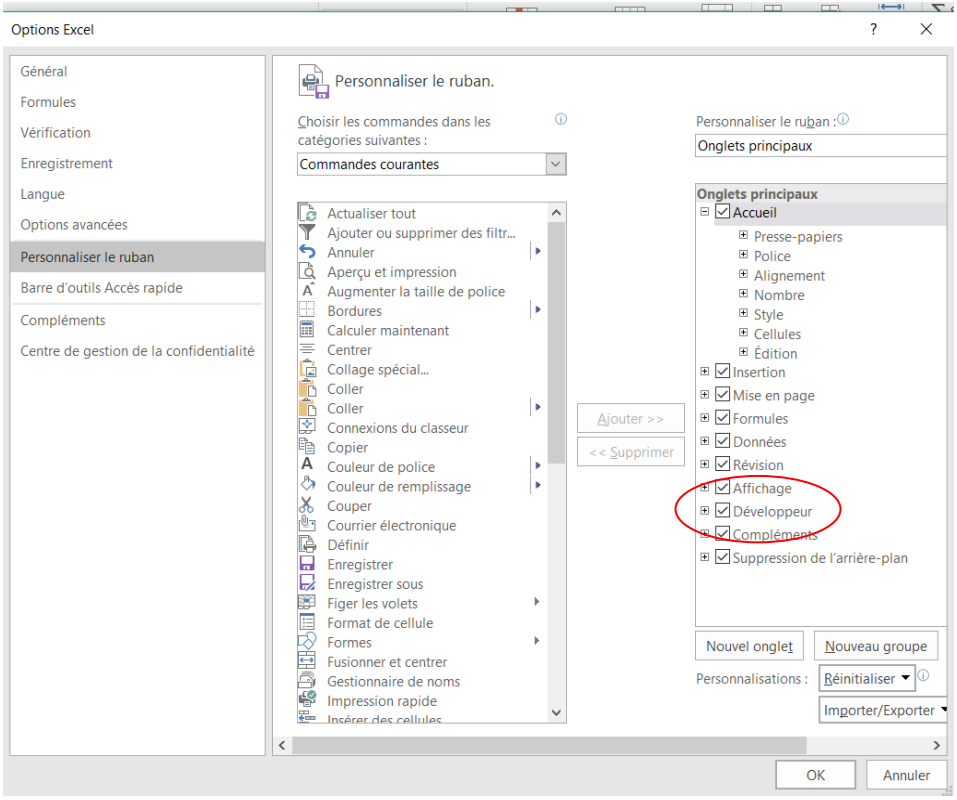
Le principal avantage de VBA est sa simplicité. Il permet au non-spécialiste, après un rapide apprentissage, de réaliser des applications complexes, en obtenant en particulier une interface de qualité. De plus, VBA permet de programmer sur toutes les applications d'Office.

Cependant, il faut garder à l'esprit que les programmes développés ne pourront être utilisés sans l'application sous laquelle ils ont été développés.

2. Création d'une macro enregistrée

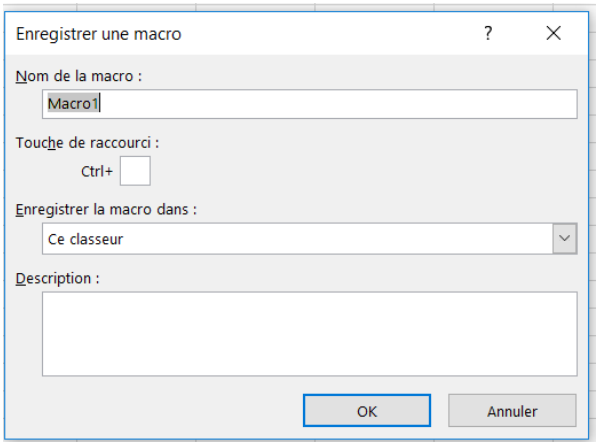
Pour accéder à VBA, vous devez afficher l'onglet développeur dans EXCEL.

- Lancez **Excel**
- Cliquez sur le menu **Fichier** et sélectionnez **Options**
- Sélectionnez l'onglet **Développeur**



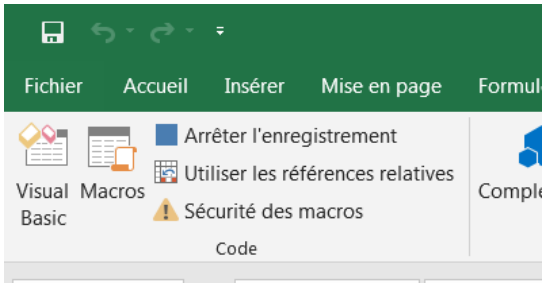
Une macro-commande est une série de commandes qui sont exécutées l'une après l'autre toujours dans le même ordre. Elles sont très pratiques pour automatiser certaines tâches répétitives. L'exercice qui suit va vous permettre de créer une macro commande. Il s'agit d'écrire Bonjour dans une cellule en gras et en italique.

- Dans le menu **Développeur**, cliquez sur **Enregistrer une Macro**



- Dans la zone **Nom de la macro**, remplacez Macro1 par **GrasItalique**
- Conservez dans la zone **Enregistrer la macro dans** : **ce classeur**
- Cliquez sur **OK** pour valider la création de la macro GrasItalique

Dans la barre d'outils l'onglet **Enregistrer une macro** s'est transformé en **Arrêt l'enregistrement**.
Ce bouton **Arrêter l'enregistrement** permet d'interrompre l'enregistrement de la macro.



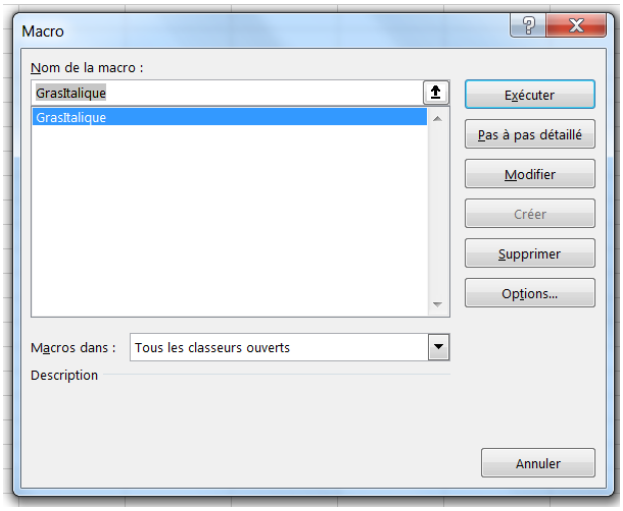
- Sélectionnez la cellule **A1**
- Tapez **Bonjour** dans la cellule A1 et validez
- Sélectionnez la cellule **A1**
- Cliquez sur le bouton droit de la souris, choisissez le menu **Format de cellule** et activez l'onglet **Police** de la boîte de dialogue **Format de cellule**
- Dans la zone **Style**, sélectionnez **Gras Italique**, puis cliquez sur **OK**
- Cliquez sur le bouton **Arrêter l'enregistrement**

Les commandes de la macro GrasItalique se sont enregistrées automatiquement.

3. Exécution d'une macro

Vous allez écrire Bonjour en GrasItalique dans une cellule quelconque.

- Effacez le contenu de la cellule **A1**
- Sélectionnez la cellule **A1**
- Dans la barre d'outils **Développeur**, sélectionnez **Macro**
- Sélectionnez **GrasItalique** dans la liste des macros disponibles
- Cliquez sur le bouton **Exécuter**



Excel applique à la cellule A1 toutes les actions que nous avons enregistrées dans la macro GrasItalique.

- Effacez le contenu de la cellule **A1**
- Sélectionnez la cellule **B1**
- Exécutez la macro **GrasItalique** comme précédemment

Excel applique les actions enregistrées à la cellule A1 et non à la cellule B1.

4. Structure et modification d'une macro

Lors de l'enregistrement de la macro, les actions ont été converties en langage Visual Basic. Pour en visualiser la syntaxe :

- Choisissez dans la barre d'outils **Développeur**, la commande **Macro**.
- Dans la boîte de dialogue choisissez la macro GrasItalique dans la liste des macros
- Cliquez sur le bouton **Modifier**

L'environnement de développement intégré d'Office, Visual Basic Editor, s'ouvre sur la fenêtre de code de votre macro.

***Sub** GrasItalique()*	**commande de début de la macro**
'	
' GrasItalique Macro	
' Macro enregistrée par Jean-Marc DESTRAC	Lignes de commentaires commençant par '
'	
Range("A1").Select	Sélection de la cellule A1
ActiveCell.FormulaR1C1 = "Bonjour"	Ecriture de Bonjour dans la cellule active
Range("A1").Select	Sélection de la cellule A1
With Selection.Font	***Début de l'instruction Avec***
.Name = "Arial"	Nom de la police de caractères
.FontStyle = "Gras italique"	Style de la police de caractères
.Size = 10	Taille de la police de caractères
.Strikethrough = False	Case barrée
.Superscript = False	Case exposant
.Subscript = False	Case indice
.OutlineFont = False	Mise en forme relief (dépend de la police choisie)
.Shadow = False	Mise en forme ombrée (dépend de la police choisie)
.Underline = xlUnderlineStyleNone	Soulignement
.ThemeColor = xlThemeColorLight1	Couleur automatique
.TintAndShade = 0	Aucune teinte et nuance
.ThemeFont = xlThemeFontMinor	Thème de la police
End With	***Fin de l'instruction Avec***
End Sub	**commande de fin de la macro**

L'expression *Selection.Font* indique à la macro qu'il s'agit d'appliquer un format de police aux cellules sélectionnées.

Entre les lignes de commandes *With* et *End With* sont définies les propriétés de l'objet *Font*.

A chaque propriété est affectée une valeur :

Booléenne : *False* (faux) ou *True* (vrai)
Chaîne de caractères : *"Gras italique"*
Numérique : 10
Constante : valeur prédéfinie qui permet de paramétrer une propriété *xlAutomatic*

Nous pouvons modifier le code de la macro pour :
- Enlever les lignes inutiles
- Pouvoir écrire Bonjour dans n'importe quelle cellule en police de taille 14

Les lignes de commentaires sont des indications ajoutées dans le code d'un programme et sont destinées à en faciliter la reconnaissance et/ou la compréhension.

```
Sub GrasItalique()
'
' GrasItalique Macro
' Macro enregistrée  par Jean-Marc DESTRAC
'
   ActiveCell.FormulaR1C1 = "Bonjour"

   With Selection.Font
      .Name = "Arial"
      .FontStyle = "Gras italique"
      .Size = 14
   End With
End Sub
```

- Enregistrez le fichier Excel et nommez le **Macro1** dans votre répertoire de travail

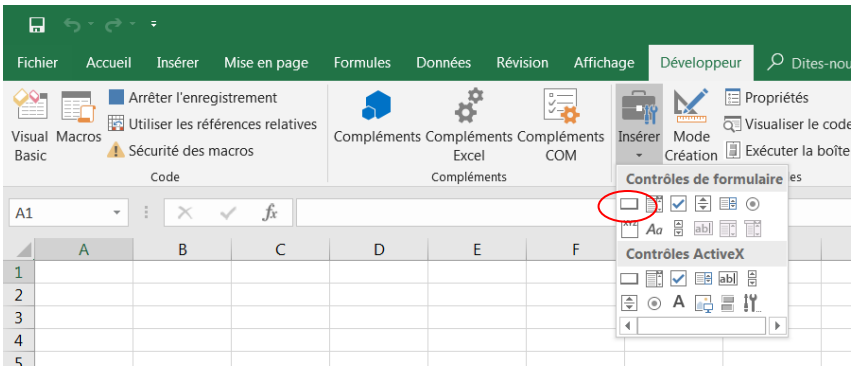
La macro sera enregistrée dans le fichier Macro1 mais ne sera disponible que si ce fichier est ouvert.

5. Affecter une macro à un bouton de commande

Vous pouvez faciliter l'accès à vos macros en les affectant à des boutons de commande que vous pouvez positionner dans votre classeur à l'endroit de votre choix.

Les contrôles sont des objets graphiques que vous placez dans un formulaire pour afficher ou entrer des données, effectuer une action ou simplifier la lecture du formulaire. Il peut par exemple s'agir de zones de texte, de zones de listes, de cases d'options, de boutons de commandes. Les contrôles permettent aux utilisateurs de sélectionner des options ou de cliquer sur des boutons qui exécutent des macros.
Microsoft Excel dispose de deux types de contrôles.

- **Les contrôles de Formulaire** sont compatibles avec les versions antérieures d'Excel, à partir de la version 5.0.
- **Les contrôles ActiveX** : tels qu'une case à cocher ou un bouton, qui proposent des options aux utilisateurs ou exécutent des macros d'automatisation de tâches.

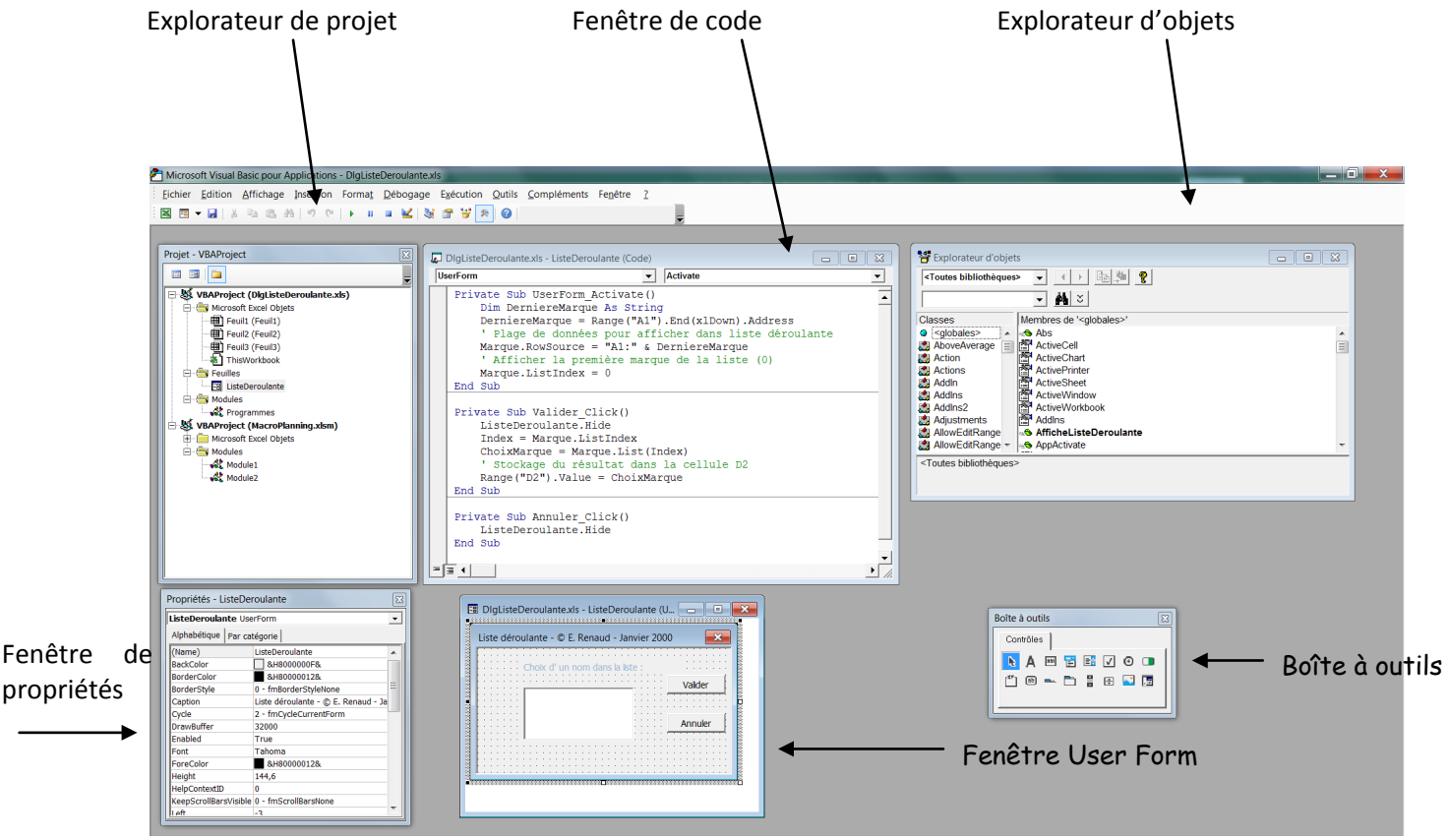


- Choisissez dans la barre d'outils **Développeur**, la commande **Insérer**
- Cliquez sur l'icône **Bouton** des **contrôles de formulaire**
- Cliquez et maintenez le bouton de la souris enfoncé tout en faisant glisser le curseur. Excel crée un bouton et affiche la boîte de dialogue **Affecter une macro.**
- Sélectionnez la macro voulue et validez la création du bouton de commande.
- Sélectionnez le texte **Bouton1** dans le bouton de commande et renommez-le Bonjour Gras Italique

6. L'environnement Visual Basic Editor

Lorsque vous choisissez de modifier une macro existante ou de créer une nouvelle macro, vous accédez à la fenêtre de code VBA. Vous pouvez accéder à cette fenêtre de code sans passer par la commande Macros d'Excel.

- Choisissez dans la barre d'outils **Développeur**, la commande **Visual Basic.**



L'explorateur de projets	permet de visualiser les différents projets et éléments constitutifs qui les composent (objets, modules, feuilles, formulaires), et d'accéder à ces éléments ou au code qui leur est attaché.
La fenêtre de Propriétés	permet de visualiser et de modifier l'ensemble des propriétés associées aux objets constitutifs d'un projet.
La fenêtre de code	permet de visualiser et de modifier les lignes de codes des macros développées dans le projet.
La fenêtre User Form	permet de créer les feuilles VBA (formulaires utilisateurs) qui serviront d'interface avec l'utilisateur.
La boîte à outils	permet de créer les différents contrôles (case à cocher, zone déroulante ….) utilisés dans les formulaires pour communiquer avec l'utilisateur.
L'explorateur d'objets	référence les propriétés, méthodes et évènements disponibles dans les bibliothèques ansi que les différentes procédures.

7. Messages et boîtes de dialogue

L'affichage de messages ou l'utilsation de boîtes de dialogue pendant l'exécution d'un programme permet de renseigner l'utilisateur sur son déroulement, ou de lui demander des informations qui en modifieront le cours.

7.1 Fonction MsgBox

Cette fonction permet d'afficher une boîte de dialogue présentant un message et de boutons de commande. Elle s'utilise selon la syntaxe suivante :

MsgBox (prompt, buttons, Title)

Prompt correspond au message affiché dans la boîte de dialogue
Buttons détermine les boutons affichés dans la boîte de dialogue (facultatif)
Title correspond au titre de la boîte de dialogue (facultatif)

- Choisissez la barre d'outils **Développeur**, la commande **Visual Basic**
- Dans la fenêtre de code en dessous de la macro GrasItalique, tapez la macro suivante :

```
Sub MessagesBoîtes()
'
'Macros d'utilisation MsgBox
'
MsgBox ("Date et heure actuelle : " & Now)
End Sub
```

- Positionnez le curseur sur une ligne de la macro et choisissez la commande **Exécuter Sub/UserForm** du menu **Exécution**

La fonction *Now* sert à afficher la date et l'heure actuelle à la suite du texte entre guillemets.
Le bouton **OK** s'affiche par défaut dans la boîte de dialogue

- Insérez la ligne suivante dans votre macro avant la commande End Sub

```
réponse = MsgBox("Voulez-vous quitter l'utilitaire ? ", vbYesNo + vbQuestion, "Sortie")
```

L'argument vbYesNo affiche les boutons oui et non
L'argument vbQuestion affiche le symbole question

- Positionnez le curseur sur une ligne de la macro et choisissez la commande **Exécuter Sub/UserForm** du menu **Exécution**

La fonction renvoie dans la variable *réponse* une valeur numérique en fonction du bouton sur lequel clique l'utilisateur.

Bouton	Constante	Valeur retournée
OK	vbOK	1
Annuler	vbCancel	2
Abandonner	vbAbort	3
Réessayer	vbRetry	4
Ignorer	vbIgnore	5
Oui	vbYes	6
Non	vbNo	7

Possibilités d'affichage des boutons et symboles :

Boutons	Constante
OK et Annuler	vbOKCancel
Abandonner, Réessayer et Ignorer	vbAbortRetryIgnore
Oui, Non et Annuler	vbYesNoCancel
Oui et Non	vbYesNo
Symboles	
Message critique	vbCritical
Question	vbQuestion
Stop	vbExclamation
Information	vbInformation

- Insérez les lignes suivantes dans votre macro avant la commande End Sub

If réponse = 7 Then
MsgBox ("Vous n'allez pas quitter l'utilitaire")
Else
MsgBox ("Au revoir, vous allez quitter l'utilitaire")
End If

Si la fonction MsgBox retourne la valeur 7 dans la variable réponse (L'utilisateur a cliqué sur Non) alors s'affiche « *Vous n'allez pas quitter l'utilitaire »,* sinon, s'affiche *« Au revoir, vous allez quitter l'utilitaire ».*

- Positionnez le curseur sur une ligne de la macro et choisissez la commande **Exécuter Sub/UserForm** du menu **Exécution**

Lorsque s'affiche la boîte de dialogue, par défaut, le premier bouton est actif. Si vous souhaitez rendre actif un autre bouton, il suffit d'insérer l'argument vbDefaultButton2 ou vbDefaultButton3 ou vbDefaultButton4.

- Modifiez la ligne suivante dans votre macro

réponse = MsgBox("Voulez-vous quitter l'utilitaire ? ", vbYesNo + vbQuestion + vbDefaultButton2, "Sortie")

Quand vous exécutez la macro, le bouton Non est actif.

7.2 Fonction InputBox

Cette fonction permet d'afficher une boîte de dialogue contenant une zone de texte légendée permettant à l'utilisateur de saisir des informations. Elle s'utilise selon la syntaxe suivante :

InputBox (prompt, Title, Default)

Prompt correspond au message affiché dans la boîte de dialogue
Title correspond au titre de la boîte de dialogue (facultatif)
Default correspond au texte affiché dans la zone de texte (facultatif)

- Dans la fenêtre de code, tapez la macro suivante :

```
Sub nom()
'
'Macros d'utilisation InputBox
'

Utilisateur = Application.UserName
votrenom = InputBox("utilisateur : " & Utilisateur & Chr(10) & "Veuillez saisir votre nom", "saisie du nom")
MsgBox ("votre nom est : " & votrenom)
End Sub
```

La première ligne permet de placer dans la variable *utilisateur* le nom d'utilisateur référencé dans Excel à l'aide de la propriété *UserName*.
Le symbole & permet d'ajouter les chaînes de caractères.
La fonction *Chr*(10) permet d'afficher la suite du texte à la ligne.
La fonction ***InpuBox*** retourne dans la variable *votrenom* le texte saisi par l'utilisateur.

- Positionnez le curseur sur une ligne de la macro et choisissez la commande **Exécuter Sub/UserForm** du menu **Exécution**

8. Ecriture et déplacement dans les cellules

8.1 Propriété Cells

Pour écrire dans une cellule de la feuille active, on peut utiliser la propriété ***Cells***. Le code qui suit écrit Bonjour dans la 4ème ligne, 2ème colonne de la feuille courante du classeur courant (cellule B4).

Cells(4, 2) = "Bonjour"

L'application de la méthode ***select*** à l'objet ***Cells*** permet de sélectionner l'ensemble des cellules de la feuille active.

Cells.Select

8.2 Propriété Range

Une autre manière de placer du texte ou du chiffre dans une cellule particulière est la propriété ***Range*** qui permet de sélectionner une cellule ou une plage de cellules :

Range("A5") = "Bonjour"

Pour attribuer une valeur à une cellule particulière qui fait partie d'une plage de cellule, on peut utiliser la syntaxe suivante qui permet d'attribuer une valeur à la 3ème cellule d'une plage :

Range("A5:A10")(3) = "Bonjour"

Le code écrit *Bonjour* dans la 3ème cellule à partir de la première cellule sélectionnée. La première est donc A5, la 2ème A6, et la 3ème A7. Excel écrit donc dans A7.

8.3 Méthode Font.color

Pour changer la couleur du texte d'une cellule et le mettre en rouge (Le code du rouge est 255), nous pouvons utiliser la méthode ***Font.color*** :

Range("A5") = "Bonjour"
Range("A5").Font.color = 255

Le code écrit "Bonjour" dans la cellule A5, et l'écrit en Rouge.

8.4 Méthode Clear

Pour effacer des informations dans les cellules, on utilise la méthode ***clear*** :

Cells.Clear	Efface le contenu des cellules de la feuille active
Cells (1,1).clear	Efface le contenu de la cellule (1,1)
Range(''A5 :D10'').clear	Efface le contenu de la table (A5 :D10)

8.5 Propriété Sheets

Cette propriété permet de sélectionner une feuille de calcul :

Sheets("Nom de la feuille de calcul").Range("B6") = "Je suis dans un autre feuille"

8.6 Méthode Activate

Cette méthode permet d'activer un autre classeur Excel déjà ouvert :

Windows("AutreFichier.xlsm").Activate
Sheets("démo").Range("B11") = "Autre classeur, Autre feuille"

Ce code écrit "Autre Classeur, Autre feuille", dans la cellule B11, de la feuille démo, du classeur déjà ouvert AutreFichier.xlsm.

8.7 Méthodes Open,Close et Save

Voici le code généré automatiquement par l'enregistreur de Macros. Il s'agit d'ouvrir le fichier, se placer dans le bon onglet, la bonne cellule, et d'y écrire un texte :

Workbooks.Open Filename:="C:\Licence\test.xlsm"	ouvre le fichier test.xlsm
Sheets("Démo").Select	sélectionne la feuille démo
Range("A1").Select	sélectionne la cellule A1
ActiveCell.FormulaR1C1 = "Bonjour"	écrit Bonjour dans la cellule active
ActiveWorkbook.Save	sauvegarde le fichier actif
ActiveWindow.Close	ferme le classeur actif

On peut remplacer les 4 premières lignes par les 2 lignes suivantes :
Workbooks("C:\Licence\test.xlsm"). Sheets("Démo").Activate
Range("A1") = "Bonjour"

Méthodes pour se déplacer dans une feuille

Selection.End(xlDown).Select	sélectionne la case la plus en bas
Selection.End(xlToRight).Select	sélectionne la case la plus à droite
Selection.End(xlToLeft).Select	sélectionne la case la plus à gauche
Selection.End(xlUp).Select	sélectionne la case la plus haute
ActiveCell.Offset(1, 0).Select	décale la sélection d'une case vers le bas
ActiveCell.Offset(-1, 0).Select	décale la sélection d'une case vers le haut
ActiveCell.Offset(0, -1).Select	décale la sélection d'une case vers la gauche
ActiveCell.Offset(0, 1).Select	décale la sélection d'une case vers la droite

La méthode Offset permet d'accéder à une cellule ou une plage de cellules située à un certain endroit (en haut, en bas, à gauche ou a droite d'une certaine cellule). La syntaxe est :
Offset (NombreDeLignesVersLeBas , NombreDeColonnesVersLaDroite)

Range("C6").Offset(2, 1) = 10

Ce code écrit le nombre 10 dans la cellule 2 lignes plus bas et 1 colonne plus à droite de la cellule C6.

8.8 Propriétés Rows et Columns

Ces propriétés permettent de sélectionner des lignes ou des colonnes entières.

Rows (''5 :10'').select	sélectionne les lignes 5 à 10
Rows (''5 :5'').select	sélectionne la ligne 5
Columns (''B :E'').select	sélectionne la colonnes B à E
Columns (''B :B'').select	sélectionne la colonnes B
Columns (''A :A, C :E, G :G'').select	sélectionne les colonnes A , de C à E et G

8.9 Propriété UsedRange

Cette propriété permet de sélectionner la plage de cellules qui englobe toutes les données contenues dans les cellules de la feuille active.
ActiveWorkbook.ActiveSheet.UsedRange.Select

9. Gestion des feuilles de calcul

Les feuilles de calcul existantes dans un classeur sont numérotées de 1 à *Sheets.Count*, et on peut y accéder en utilisant leur nom ou leur numéro d'ordre :

Sheets.Count	Donne le nombre de feuilles existantes
Sheets(1).Name	Donne le nom de la feuille la plus à gauche
Sheets(Sheets.Count).Name	Donne le nom de la feuille la plus à droite
ActiveSheet.Name	Donne le nom de la feuille active
ActiveSheet.Index	Donne la position la feuille active (3 si c'est la 3ème à partir de la gauche)

9.1 Effacement des feuilles

On utilise la méthode *Delete* pour effacer les feuilles.

Sheets("feuille à effacer").Delete

Afin d'éviter le message d'erreur d'Excel "Attention, les feuilles sélectionnées vont être effacées", il faut envoyer la touche ENTER à l'aide de Sendkeys, avec la syntaxe suivante :

SendKeys ("{ENTER}")
Sheets("Onglet inutile").Delete

9.2 Création des feuilles de calcul

On utilise la méthode *Add* pour ajouter les feuilles.

Sheets.Add	ajoute un onglet juste avant l'onglet actif, dont le nom est FeuilX
Sheets.Add.Name	donne le nom de l'onglet que l'on vient de créer
Sheets.Add Worksheets(1)	ajoute une feuille avant la première du classeur
Sheets.Add , Worksheets(WorkSheets.Count)	ajoute une feuille après la dernière du classeur
Sheets.Add , Worksheets(WorkSheets.Count), 3	insère de 3 feuilles après la dernière feuille
Sheets.Add.Name = "Nouvelle"	insère une feuille et la renomme *Nouvelle*

9.3 Copie des feuilles existantes

La copie de feuille se passe exactement de la même manière que la création, mais on utilise la méthode ***Copy*** à la place de la méthode *Add*

Sheets("Nom").Copy	Copie la feuille Nom juste avant elle-même

9.4 Déplacement des feuilles existantes

On utilise la méthode ***Move :***

Sheets("Nom").Move, Worksheets(WorkSheets.Count)	Déplace la feuille Nom avant la 1ère feuille
ActiveSheet.Name = "nouveau nom"	Renomme la feuille active

10. Instructions de décision

10.1 If.....Then.....Else.....End If

Cette instruction permet d'exécuter une ou plusieurs instructions selon le résultat d'une condition.

- Dans le fichier macro1.xlsm, rajoutez la procédure suivante :

```
Sub Condition()
  If Date < DateValue("15/09/2017") Then
    Range("A1") = Date
    MsgBox "Nous sommes le " & Format(Date, "dddd d mmmm yyyy")
  End If
End Sub
```

- Exécutez la procédure.

Si la condition est respectée, les deux lignes de commandes s'exécutent, sinon, il ne se passe rien.

- Complétez la procédure :

```
Sub Condition()
  If Date < DateValue("15/09/2017") Then
    Range("A1") = Date
    MsgBox "Nous sommes le " & Format(Date, "dddd d mmmm yyyy")
  Else
    MsgBox "Nous avons dépassé le 15 septembre 2017"
  End If
End Sub
```

- Exécutez la procédure.

Si la condition est respectée, les deux lignes de commandes s'exécutent, sinon, la ligne de commande située entre Else et End If s'exécute.

Vous pouvez imbriquer plusieurs If.....Then.....Else autant de fois que vous le désirez, mais attention à la façon dont vous les écrivez, elles doivent se présenter ainsi :

```
If condition1 then
 instructions ou pas
   If condition2 then
bloc d'instructions qui s'exécutent la conditon1 et condition2 sont vraies
 End if
  instructions ou pas
End if
```

- Dans le fichier macro1.xls, rajoutez la procédure suivante :

```
Sub Conditionimbriquées()
  UneDate = DateValue("15/09/2017")
  NbJours = Date - UneDate
  If NbJours > 0 Then
    MsgBox NbJours & " jours se sont déroulés depuis le " & Format(UneDate, "dddd d mmmm yyyy")
          ElseIf NbJours = 0 Then
                MsgBox "Nous sommes le " & Format(UneDate, "dddd d mmmm yyyy")
           Else
          MsgBox "Encore " & -NbJours & " jours avant le " & Format(UneDate, "dddd d mmmm yyyy")
  End If
End Sub
```

- Exécutez la procédure.

10.2 Select Case

Cette instruction permet permet d'envisager différentes valeurs pour une même expression et de déterminer des instructions spécifiques pour chaque cas envisagé. Elle évite l'utilisation de ElseIf.

- Dans le fichier macro1.xlsm, rajoutez la procédure suivante :

```
Sub ConditionSelectCase()
   For N = 1 To 10
      Range("A" & N) = N
   Next
   For Each MaCel In Range("A1:A10")
      Select Case MaCel
         Case 2
            MaCel.Offset(0, 1) = MaCel + 1000
         Case 1, 4
            MaCel.Offset(0, 1) = MaCel + 10000
         Case 5 To 6
            MaCel.Offset(0, 1) = MaCel + 10
         Case Is > 7
            MaCel.Offset(0, 1) = MaCel + 100
         Case Else
            MaCel.Offset(0, 1) = MaCel
      End Select
   Next
End Sub
```

- Exécutez la procédure.

11. Instructions de boucle

11.1 Do.....Loop

Ces instructions permettent de répéter un bloc d'instructions indéfiniment :

- Jusqu'à ce qu'une condition soit vraie, vous utilisez alors **Until** devant la condition
- Tant qu'une condition est vraie, vous utilisez alors **While** devant la condition.

Vous pouvez obliger votre code à exécuter le bloc d'instruction au moins 1 fois, dans ce cas les instructions se présentent ainsi :

Jusque	**Tant que**
Do Bloc d'instructions ***Loop Until*** conditions	***Do*** Bloc d'instructions ***Loop While*** conditions

Pour vérifier la condition dès le début dans la boucle, elle peut donc ne pas être exécutée une seule fois, utilisez les instructions ainsi :

Jusque	**Tant que**
Do Until conditions Bloc d'instructions ***Loop***	***Do While*** conditions Bloc d'instructions ***Loop***

- Dans le fichier macro1.xlsm, rajoutez la procédure suivante :

```
Sub BoucleLoop()
    Do
    N = N + 1
    MsgBox "Message de test " & N
    Loop Until N = 5
End Sub
```

- Exécutez la procédure. Le message s'affiche 5 fois

- Complétez la procédure :

```
Sub BoucleLoop()
    Do
    N = N + 1
    MsgBox "Message de test " & N
    If N = 2 Then Exit Do
    Loop Until N = 5
End Sub
```

- Exécutez la procédure. Le message ne s'affiche que 2 fois

11.2 For.....Next

Ces instructions permettent de répéter un bloc d'instructions un nombre de fois déterminé dans le code. Vous indiquez la valeur de départ du compteur, la valeur de fin de ce compteur et éventuellement le pas d'incrémentation de cette valeur. La syntaxe est la suivante :

***For** compteur = x **To** y **Step** Pas*

- Dans le fichier macro1.xls, rajoutez la procédure suivante :

```
Sub BoucleForNext()
    For N = 1 To 3
    MsgBox "Message de test numéro " & N
    Next
End Sub
```

- Exécutez la procédure. Le message s'affiche 3 fois

Par défaut la valeur s'incrémente de 1, si vous voulez l'incrémenter différemment ou même la décrémenter utilisez l'argument **Step** :

Avec le code ***For** N = 1 **To** 5 Step 2*, la variable N prend les valeurs 1,3 et 5.

Avec ***For** N = 5 **To** 1 **Step** -2*, la variable N prend les valeurs 5,3 et 1.

Vous pouvez sortir prématurément d'une boucle en utilisant ***Exit For***.

Vous pouvez imbriquer plusieurs boucles, mais ces boucles doivent s'imbriquer et non se superposer. Pour éviter les erreurs, vous pouvez ajouter le nom de la variable après le mot ***Next***.

- Dans le fichier macro1.xlsm, rajoutez la procédure suivante :

```
Sub BoucleForimbriquées()
    For Col = 1 To 2
        For Ligne = 1 To 5
        Cells(Ligne, Col) = Ligne & " * " & Col
        Next Ligne
        MsgBox "Colonne n° " & Col & " est remplie"
    Next Col
End Sub
```

- Exécutez la procédure

11.3 For Each.....Next

Ces instructions permettent de généraliser un traitement à l'ensemble des objets d'une collection et s'utilise selon la syntaxe suivante :

```
For Each élément In Collection
    Instructions
Next élément
```

- Dans le fichier macro1.xlsm, rajoutez la procédure suivante :

```
Sub BoucleForEach()
        For N = 1 To 10
                Cells(N, 1) = 997 + N
         Next
Range("A1:A10").Select
        For Each ContenuCellule In Selection.Cells
                If ContenuCellule >= 1000 And ContenuCellule =< 1005 Then
                ContenuCellule.Font.ColorIndex = 3
                End If
        Next ContenuCellule
End Sub
```

- Exécutez la procédure

La première partie de la procédure écrit dans les 10 premières cellules des valeurs allant de 998 à 1007. La boucle **For Each** permet de mettre en rouge les valeurs comprises entre 1000 et 1005.

12. Gestions des erreurs

Lorsqu'une erreur survient, un message s'affiche, celui-ci est difficile de compréhension pour l'utilisateur, mais de plus la procédure s'arrête, ce qui peut poser de gros problèmes pour votre utilitaire. Vous devez tout d'abord tester votre code le plus possible pour corriger les erreurs. Vous pourrez ainsi ajouter le code adéquat pour éviter une erreur, par exemple interdire une division si le diviseur est égal à 0. Mais VBA vous permet également de détecter les erreurs et d'effectuer certaines actions pour y remédier.

En ajoutant un gestionnaire d'erreurs, vous pourrez soit remplacer les messages d'erreurs peut explicite pour l'utilisateur, soit remettre l'application en état de travail correct, soit corriger l'erreur puis reprendre le cours de votre procédure.

Instruction **On Error**

Elle permet de créer un gestionnaire d'erreur et elle se déclenche lorsqu'une erreur survient.
Elle s'utilise selon la syntaxe suivante :

```
Sub MaProcédure()
On Error GoTo GestionErreur
    'début du code
    '....
    'Fin du code
    Exit Sub
GestionErreur:
    'code de gestion d'erreur
    'avec retour dans le code après gestion ou sortie de la procédure
End Sub
```

En cas d'erreur le code suivant l'étiquette *GestionErreur* est exécuté.
Il est important de même l'instruction ***Exit Sub*** avant l'étiquette indiquant le début du code pour les erreurs, sinon votre procédure exécute ses lignes automatiquement.

Str(Err) vous donne le numéro d'erreur, celui qui s'affiche dans une fenêtre vous proposant le débogage, si "On Error GoTo NomEtiquette" n'existe pas.
Error(Err) vous donne le texte correspondant à l'erreur.

Pour retourner dans le code normal après traitement de l'erreur, vous pouvez utiliser :

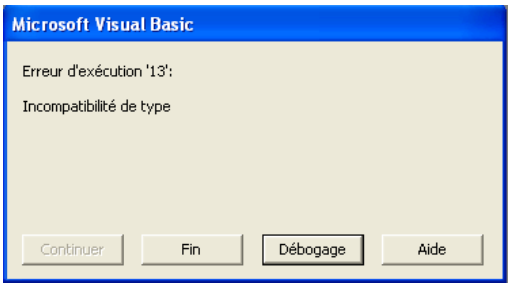
- ***Resume*** qui renvoie sur la ligne d'instruction qui a provoqué l'erreur
- ***Resume Next*** qui renvoie sur la ligne d'instruction qui suit celle qui a provoqué l'erreur

- Dans le fichier macro1.xls, rajoutez la procédure suivante :

```
Sub Rechercher_Dimanche()
  Do
  date_saisie = InputBox(msg + "Entrez une date !", "Rechercher un dimanche")
  msg = "le " + date_saisie + "n'est pas un dimanche " + Chr(10)
  Loop While Weekday(date_saisie) <> 1
  MsgBox "le " + date_saisie + " est un dimanche.", vbInformation,"Rechercher un dimanche"
End Sub
```

Exécutez la procédure

La fonction ***Weekday*** renvoie 1 si la date saisie est un dimanche, 2 si la date est un lundi etc…
La boucle ***Do …Loop While*** permet d'afficher la boîte de dialogue tant que la date saisie n'est pas un dimanche.
Si vous saisissez un texte à la place d'une date, la fonction ***Weekday*** renvoie l'erreur suivante :



Pour gérer cette erreur N°13 modifiez la procédure comme suit :

```
Sub Rechercher_Dimanche()
On Error GoTo erreur_rechercher_dimanche
début:
   Do
   date_saisie = InputBox(msg + "Entrez une date !", "Rechercher un dimanche")
   msg = "le " + date_saisie + "n'est pas un dimanche " + Chr(10)
  Loop While Weekday(date_saisie) <> 1
  MsgBox "le " + date_saisie + " est un dimanche.", vbInformation,"Rechercher un dimanche"
Exit Sub
erreur_rechercher_dimanche:
   If Err = 13 Then
         msg = "La dernière saisie n'est pas une date !" + Chr(10)
   Resume début
         Else
         MsgBox "Erreur n° : " + Str(Err) + " Erreur : " + Error(Err)
   Resume début
   End If
End Sub
```

13. Gestions des Procédures et Fonctions

VBA permet d'utiliser 2 types de procédures :

- La procédure ***Sub . . . . . End Sub*** est structurée de la manière suivante :

Sub *NomDeLaProcédure()*
Instructions
End Sub

Elle exécute une série d'instructions sans renvoyer de valeur comme vu précédemment.

- La procédure ***Function . . . . . End Function*** est structurée de la manière suivante :

Function *NomDeLaProcédure(Arguments)*
Instructions
. . . . . .
NomDeLaProcédure = Expression
End Function

Elle exécute une série d'instructions en renvoyant une valeur.

NomDeLaProcédure = Expression affecte une valeur à la fonction pour la renvoyer à la procédure appelante.

- Dans le fichier macro1.xlsm, rajoutez la procédure et la fonction suivante :

Sub *MaProcédure()*
Rayon = 10
MsgBox "La surface du cercle est de " & ***SurfaceCercle(Rayon)*** *& " centimètres carrés.", vbOKOnly & vbInformation, "Appel de fonction"*
End Sub

Function *SurfaceCercle(Rayon)*
Pi = 3.14
*SurfaceCercle = Pi * Rayon * Rayon*
End Function

- Exécutez la procédure