Chapitre 1

Les Bases des Macros et des Fonctions

1.1 Introduction à VBA

VBA (Visual Basic for Applications) est un langage de programmation intégré dans les applications Microsoft (comme Excel, Word, etc.), qui permet d'automatiser des tâches et de créer des macros ou des fonctions personnalisées. Il est particulièrement utilisé dans Excel pour automatiser des calculs, manipuler des données ou créer des interfaces personnalisées.

1.2 Structure d'un Code VBA

Un code VBA est généralement constitué de deux éléments principaux : les **Subroutines** (ou macros) et les **Fonctions**.

1.2.1 Les Macros (Subroutines)

Une **macro** est une séquence d'instructions qui est exécutée lorsque vous l'appelez. Les macros sont généralement utilisées pour automatiser des tâches répétitives. Elles n'ont pas de valeur de retour, ce qui signifie qu'elles effectuent des actions sans renvoyer de résultat.

Exemple de macro :

```
1         Sub macro1()
2         MsgBox "Hello !!"
3
4         'instruction1
5         'instruction2
6         End Sub
7
```

Listing 1.1 – Macro simple

— **Sub macro1()** : Cela définit une macro appelée `macro1`.
— **MsgBox "Hello !!"** : Cette instruction affiche une boîte de message avec le texte `"Hello !!"`.
— **'instruction1** et **'instruction2** : Ce sont des commentaires. En VBA, tout ce qui suit un apostrophe (') est ignoré lors de l'exécution du code.
— **End Sub** : Cela marque la fin de la macro.

1.2.2 Les Fonctions

Une **fonction** est similaire à une macro, mais elle permet de **retourner une valeur**. Les fonctions sont souvent utilisées pour effectuer des calculs ou manipuler des données et renvoyer un résultat.

Exemple de fonction :

```
1         Function mafonction1()
2         'instruction1
3         'instruction2
4         End Function
5
```

Listing 1.2 – Définition d'une fonction

Les fonctions peuvent également contenir des calculs, comme l'exemple ci-dessous :

```
1        Function addition(a As Integer, b As Integer)
   As Integer
2        addition = a + b
3        End Function
4
```

Listing 1.3 – Fonction avec retour de valeur

Cette fonction prend deux paramètres (`a` et `b`), les additionne, puis renvoie le résultat.

1.2.3 Différence entre `Sub` et `Function`

— **Sub (Subroutine)** : Une procédure qui effectue des actions mais ne renvoie pas de valeur.
— **Function** : Une procédure qui peut effectuer des actions et qui renvoie une valeur.

1.3 Comment Utiliser les Macros et les Fonctions

1.3.1 Exécution d'une macro

Pour exécuter une **macro** dans Excel, vous pouvez :
— Lier la macro à un bouton.
— L'exécuter directement depuis l'éditeur VBA.

1.3.2 Utilisation d'une fonction dans une cellule Excel

Une **fonction** peut être utilisée dans une cellule Excel, comme une fonction Excel standard. Par exemple, une fonction `addition` que vous avez définie peut être appelée dans une cellule de la manière suivante :

```
=addition(5, 10)
```

Cela renverra le résultat de l'addition de 5 et 10, soit 15.

1.4 Résumé

— **Sub** : Crée une macro qui exécute des actions mais ne renvoie pas de valeur.
— **Function** : Crée une fonction qui peut effectuer des actions et renvoyer une valeur.
— Les **commentaires** (lignes commençant par ') sont utilisés pour expliquer le code sans affecter son exécution.

1.5 Conclusion

Le VBA est un outil puissant pour automatiser les tâches dans les applications Microsoft, comme Excel. Vous pouvez utiliser des **macros** pour exécuter des séries d'actions et des **fonctions** pour effectuer des calculs ou manipuler des données tout en renvoyant des résultats. Les commentaires dans le code sont essentiels pour documenter et clarifier les actions sans interférer avec l'exécution.

Chapitre 2

Utilisation de la fonction MsgBox en VBA

2.1 Exemple simple de MsgBox

La fonction `MsgBox` permet d'afficher une boîte de dialogue à l'utilisateur. Voici un exemple :

```
Sub bonjour()
MsgBox "Bonjour ! Nous sommes le : " & Date
'L'instruction Date nous donne la date du jour
End Sub
```

2.2 Personnalisation de MsgBox

Vous pouvez personnaliser les boutons et les icônes de la boîte de dialogue. Exemple :

```
Sub bonjour_personnalise()
MsgBox "Bonjour ! Nous sommes le : " & Date,
vbYesNo + vbCritical, "Titre personnalisé
End Sub
```

Dans cet exemple :

— `vbYesNo` ajoute les boutons "Oui" et "Non".

— `vbCritical` affiche une icône d'alerte.

— Le titre de la boîte est défini par le troisième argument.

2.3 Retour à la ligne dans MsgBox

Pour insérer un retour à la ligne dans une boîte de dialogue, utilisez la fonction `Chr(10)` :

```
1  Sub msgbox_retour_ligne()
2  MsgBox "Bonjour !" & Chr(10) & "Nous sommes le : "
   & Date, vbOKOnly, "Message structuré"
3  End Sub
```

2.4 Confirmation avec MsgBox

Une autre utilisation fréquente de `MsgBox` est de demander une confirmation avant d'effectuer une action. Exemple :

```
1  Sub color()
2  If MsgBox("Voulez-vous appliquer la couleur rouge à
   la cellule F2 ?", vbYesNo, "Confirmation") = vbYes
   Then
3  Range("F2").Interior.Color = RGB(255, 0, 0)
4  Else
5  Range("F2").ClearFormats
6  End If
7  End Sub
```

2.5 Exemple avancé : Mise en couleur automatique

Ce code applique des couleurs et des commentaires aux cellules d'une plage en fonction de leur valeur :

```
1   Sub applicouleur()
2   If MsgBox("Voulez-vous appliquer la couleur et les
    commentaires ?", vbYesNo, "Confirmation") = vbNo Then
3   Sheets("Feuil2").Range("B2:C13").Interior.Pattern =
    xlNone
4   Sheets("Feuil2").Range("C2:C13").ClearContents
5   Exit Sub
6   End If
7
8   For i = 2 To 13
9   If Sheets("Feuil2").Range("B" & i).Value > 0 Then
10  Sheets("Feuil2").Range("B" & i).Interior.Color =
    RGB(0, 255, 0) ' Vert
11  Sheets("Feuil2").Range("C" & i).Value = "Positif"
12  ElseIf Sheets("Feuil2").Range("B" & i).Value < 0
    Then
13  Sheets("Feuil2").Range("B" & i).Interior.Color =
    RGB(255, 0, 0) ' Rouge
14  Sheets("Feuil2").Range("C" & i).Value = "Négatif"
15  Else
16  Sheets("Feuil2").Range("B" & i).Interior.Color =
    RGB(0, 0, 255) ' Bleu
17  Sheets("Feuil2").Range("C" & i).Value = "Nul"
18  End If
19  Next i
20  End Sub
```

Chapitre 3

Gestion des erreurs et affichage des informations d'un pays

Ce chapitre explore des concepts avancés en VBA liés à la gestion des erreurs, à l'interaction utilisateur via les `InputBox` et `MsgBox`, et à la manipulation de plages de données dans Excel.

3.1 Théorie : Les notions abordées

3.1.1 La gestion des erreurs

La gestion des erreurs en VBA permet de prévenir les plantages en cas d'entrée ou d'événement inattendu. L'instruction `On Error GoTo` redirige l'exécution vers un point spécifique du code lorsqu'une erreur survient.

- `On Error GoTo [nom_du_label]` : Détermine le point d'entrée en cas d'erreur.
- `Resume Next` : Ignorer l'erreur et passer à l'instruction suivante.
- `Err.Number` : Donne le numéro de l'erreur rencontrée.
- `Err.Description` : Retourne une description de l'erreur.

Exemple : Gestion d'une erreur

```
On Error GoTo erreur
' Code risquant de générer une erreur

Exit Sub ' Sortir pour éviter d'exécuter le label en l'absence d'erreur

erreur:
MsgBox "Une erreur est survenue : " & Err.Description, vbCritical
```

3.1.2 Les interactions utilisateur

Les interactions utilisateur en VBA se font souvent à l'aide des fonctions suivantes :

— `MsgBox` : Affiche une boîte de message. Elle peut afficher des informations, poser des questions ou alerter l'utilisateur.

— `InputBox` : Permet de demander à l'utilisateur une entrée, qui sera ensuite traitée dans le programme.

Paramètres principaux de `MsgBox` :

— `Prompt` : Le texte affiché dans la boîte.

— `Buttons` : Définit les boutons et icônes (ex. `vbYesNo`, `vbCritical`).

— `Title` : Spécifie le titre de la boîte.

Exemple : Une boîte de message simple

```
MsgBox "Ceci est un message d'information.", vbInformation, "Information"
```

Paramètres principaux de `InputBox` :

— `Prompt` : Texte expliquant ce qui est attendu de l'utilisateur.

— `Title` : Titre de la boîte.

— `Default` : Valeur par défaut de l'entrée.

Exemple : Demander une valeur numérique

```
Dim valeur As Integer
valeur = InputBox("Veuillez saisir un entier :", "Entrée de données", 0)
```

3.1.3 La gestion des plages de données

En VBA, les plages de données sont manipulées à l'aide de la méthode `Range`. Voici quelques concepts clés :

— `Range("A1")` : Référence à une cellule spécifique.
— `Range("A1:B10")` : Référence à une plage.
— `Interior.Color` : Change la couleur de fond d'une cellule.
— `Value` : Récupère ou affecte une valeur à une cellule.

Exemple : Appliquer une couleur à une cellule

```
Range("A1").Interior.Color = RGB(255, 0, 0) ' Rouge
```

3.2 Exemples pratiques avec explications

3.2.1 Exemple 1 : Gestion des erreurs avec `TypeVal`

Le code ci-dessous montre comment demander une valeur numérique à l'utilisateur et gérer les erreurs de saisie.

```
Sub TypeVal()

' En cas d'erreur, aller au message d'alerte
On Error GoTo msg_erreur

' Définir la variable
Dim VarNum As Integer

' Saisie de l'utilisateur
VarNum = InputBox("Veuillez saisir une valeur numérique", _
"Type variable", 0)

' Affectation de la valeur à une cellule
Sheets("Feuil1").Range("B3").Value = VarNum

Exit Sub

msg_erreur:
MsgBox "Erreur : saisie non numérique. Veuillez réessayer.", _
vbCritical, "Alerte"
Call TypeVal ' Relance la procédure
End Sub
```

Analyse :
— Si l'utilisateur saisit une valeur invalide, le message d'alerte s'affiche et l'utilisateur doit réessayer.

— La valeur est ensuite insérée dans la cellule B3.

3.2.2 Exemple 2 : Affichage des informations d'un pays

Ce code permet d'afficher des informations spécifiques à un pays sélectionné par l'utilisateur dans une feuille Excel.

```
1   Sub FichePays()
2
3   Dim NumPays As Integer
4   Dim sh As Worksheet
5
6   On Error GoTo msg_erreur
7
8   Set sh = Sheets("DATA")
9   NumPays = InputBox("Veuillez saisir un entier entre
    2 et 6", _
10  "Sélection du pays", 2)
11
12  If NumPays >= 2 And NumPays <= 6 Then
13  MsgBox "Pays : " & sh.Range("A" & NumPays).Value &
    Chr(10) & _
14  "Capitale : " & sh.Range("B" & NumPays).Value,
    vbInformation
15  Else
16  GoTo msg_erreur
17  End If
18
19  Exit Sub
20
21  msg_erreur:
22  MsgBox "Erreur : valeur invalide. Veuillez
    réessayer.", vbCritical, "Alerte"
23  Call FichePays
24  End Sub
```

Analyse :

— L'utilisateur doit sélectionner un numéro correspondant à un pays dans la plage 2 à 6.

— Les informations du pays sont extraites de la feuille DATA et affichées dans une boîte de message.

— Si la saisie est incorrecte, une alerte s'affiche, et la procédure est relancée.

3.3 Conclusion

Ces exemples montrent comment gérer les interactions utilisateur et les erreurs dans un programme VBA. Les notions de `InputBox`, `MsgBox`, et de gestion des plages permettent de créer des applications interactives robustes. En combinant ces concepts, il est possible d'améliorer la fiabilité et l'expérience utilisateur des macros VBA.