Thème numéro 16

Thèmes essentiels de la séance :

1. **Fonctions personnalisées**
2. **Ecriture de code VBA**

Objet : Création d'une fonction personnalisée

Cette fonction portera le nom **NROMAIN**, et aura pour but de convertir en chiffres romains, sous forme de chaîne de caractères, un nombre compris entre 1 et 9999. Une telle fonction personnalisée peut être utilisée dans une feuille de calcul comme le seraient les fonctions déjà définies par EXCEL.

- Ouvrir EXCEL
- Créer un module.

Introduisez les instructions qui suivent dans ce module. (Ces instructions sont précédées, dans ce fascicule, d'un losange. Ce losange n'a pas à être introduit dans le module).

♦ `Function nromain(n)`

Cette instruction déclare une fonction nommée* NROMAIN.** ***Il existe en effet une fonction standard d'EXCEL nommée* ROMAIN *qui réalise à peu près le même travail. Il ne faut pas qu'EXCEL confonde la fonction que nous introduisons ici avec la fonction prédéfinie* ROMAIN. *Notre fonction* NROMAIN *utilise un seul paramètre* n *de type* Variant *(c'est pourquoi nous ne déclarons pas pour ce paramètre de type particulier), et rend également un type* Variant *(c'est pourquoi nous n'avons pas déclaré de type particulier pour la valeur renvoyée)

```
♦     Dim j As Integer
♦     Dim k As Integer
♦     Dim r As String
```

***Ces trois variables sont des variables de travail pour la fonction* NROMAIN.**

⇒**j** contiendra successivement le nombre de milliers, de centaines et de dizaines

⇒**k** est la variable ***compteur*** d'une boucle pour inscrire les **M** des milliers

⇒**r** est la chaîne de caractères dans laquelle nous construirons le résultat

```
♦ If TypeName(n) = "Range" Then n = n.Value
```

Lorsqu'on appelle* NROMAIN *avec pour paramètre une référence de cellule, par exemple A3, la fonction reçoit dans* n *un objet de classe* Range*, inutilisable directement dans un calcul numérique. Pour pouvoir réaliser des calculs, on extrait l'attribut* Value *de cet objet* Range*. L'attribut* Value *d'un objet* Range *peut être de type* Double*,* Boolean *ou type* String*. Lorsqu'au contraire on appelle* NROMAIN *avec un paramètre numérique, la fonction reçoit un paramètre de type* Double*.

```
♦     If Not IsNumeric(n) Then
♦         nromain = CVErr(xlErrNum)
♦         Exit Function
♦     End If
```

A ce stade,* n *est soit de type* Double*,* Boolean *ou* String*.

Si la variable* n *ne contient pas une donnée d'un type numérique (c'est-à-dire si elle contient une chaîne ou un booléen), on quitte la fonction (grâce à l'instruction* Exit Function *) après avoir affecté à son identifiant* nromain *une valeur d'erreur résultant de la conversion dans le type* Error*, par la fonction* CVErr*, de la constante prédéfinie de VBA* xlErrNum*.

***L'affectation d'une valeur à l'identifiant* nromain *de la fonction fixe la valeur rendue à l'appelant. Cette valeur d'erreur s'affichera, dans une feuille EXCEL, sous la forme* #NUM!

Vous comprenez à présent pourquoi nous avons laissé le type de la valeur renvoyée par la fonction ***nromain*** être un ***Variant***. Ceci permet à cette fonction de renvoyer aussi bien une valeur d'erreur qu'une chaîne de caractères.

```
♦     If n > 9999 Or n < 1 Then
♦         nromain = CVErr(xlErrValue)
♦         Exit Function
```

```
    End If
```

n ***contient à présent une valeur d'un type numérique. On a décidé de ne pas convertir en chiffres romains une valeur négative, nulle ou supérieure à 9999. Si la donnée ne correspond pas à ces exigences, on quitte la fonction en rendant à l'appelant (une feuille de calcul) une valeur d'erreur qui provoquera l'affichage de*** **#VALUE!**

```
    r = ""
```

Puisque la conversion de notre nombre en chiffres romains n'a pas encore commencé, le résultat de la conversion est encore une chaîne vide.

```
    'extraction milliers
    j = n \ 1000
    n = n - 1000 * j
```

La variable **j** ***contient le résultat de la division entière de*** **n** ***par*** **1000** ***(opérateur \). On obtient ainsi le nombre de milliers.***

On retranche de **n** ***le produit de*** **1000** ***par*** **j.** ***On obtient ainsi le nombre qui subsiste après élimination des milliers. Ce nombre servira dans les étapes suivantes.***

La ligne `'extraction milliers` ***est un commentaire***

```
    'conversion milliers
    For k = 1 to j
        r = r + "M"
    Next k
```

Cette boucle **For** ***introduit dans*** **r** ***autant de lettres*** **M** ***qu'il y avait à l'origine de milliers dans*** **n.**

La ligne `'conversion milliers` ***est un commentaire***

```
    'extraction centaines
    j = n \ 100
    n = n - 100 * j
```

L'extraction des centaines est analogue à celle des milliers

```
    'conversion centaines
```

```
♦     Select Case j
♦         Case 1: r = r + "C"
♦         Case 2: r = r + "CC"
♦         Case 3: r = r + "CCC"
♦         Case 4: r = r + "CD"
♦         Case 5: r = r + "D"
♦         Case 6: r = r + "DC"
♦         Case 7: r = r + "DCC"
♦         Case 8: r = r + "DCCC"
♦         Case 9: r = r + "CM"
♦     End Select
```

La valeur de* j *est comprise entre* 0 *et* 9*. Les valeurs de 1 à 9 sont traitées individuellement car il ne suffit pas d'ajouter autant de lettres* C *qu'il y a de centaines. La valeur 0, qui n'est pas citée, ne provoque aucun ajout.

```
♦     'extraction dizaines
♦     j = n \ 10
♦     n = n - 10 * j
♦     'conversion dizaines
♦     Select Case j
♦         Case 1: r = r + "X"
♦         Case 2: r = r + "XX"
♦         Case 3: r = r + "XXX"
♦         Case 4: r = r + "XL"
♦         Case 5: r = r + "L"
♦         Case 6: r = r + "LX"
♦         Case 7: r = r + "LXX"
♦         Case 8: r = r + "LXXX"
♦         Case 9: r = r + "XC"
♦     End Select
```

L'extraction et la conversion des dizaines sont en tout point analogues à celles des centaines. Seuls les textes introduits dans* r *changent.

```
♦     'conversion unités
♦     Select Case n
♦         Case 1: r = r + "I"
♦         Case 2: r = r + "II"
♦         Case 3: r = r + "III"
♦         Case 4: r = r + "IV"
♦         Case 5: r = r + "V"
♦         Case 6: r = r + "VI"
♦         Case 7: r = r + "VII"
♦         Case 8: r = r + "VIII"
♦         Case 9: r = r + "IX"
♦     End Select
```

Après l'extraction des dizaines, ne subsiste dans* n *qu'un nombre d'unités compris entre* 0 *et* 9*. C'est pourquoi l'instruction* Select Case *porte ici directement sur la variable* n*. Les textes introduits sont encore une fois différents.

♦ `nromain = r`

L'affectation de la valeur de* r *à l'identifiant de la fonction fait que c'est cette valeur qui sera rendue à l'appelant lors de l'appel de la fonction.

♦ `End Function`

Cette instruction termine le code de la fonction. Vous n'avez pas à l'introduire, elle a été introduite automatiquement par l'éditeur Visual Basic.

***Remarque : les noms des constantes prédéfinies* xlErrNum *et* xlErrValue *peuvent être trouvées en recherchant dans l'explorateur d'objets la classe* XlCVError.**

- Vous pouvez à présent sélectionner une feuille de calcul et introduire, par exemple en C3, la formule **=NROMAIN(A1)**
- En introduisant en A1 diverses valeurs, vous pourrez vérifier le traitement des différents cas d'erreurs prévus par la fonction :
  - un nombre compris entre 1 et 9999 est convenablement converti
  - une chaîne de caractères provoque en C3 l'affichage de **#NUM!**
  - un booléen tel que " VRAI " ou " FAUX " provoque en C3 l'affichage de **#NUM!**
  - un nombre négatif, nul ou supérieur à 9999 provoque en C3 l'affichage de **#VALUE!**
- Vous pourrez ensuite introduire dans une cellule quelconque une formule telle que **=NROMAIN(1234)** ou **=NROMAIN("toto")** et vérifier l'apparition des valeurs attendues.
- En cas de difficulté, introduisez un point d'arrêt sur la première instruction exécutable de la fonction :

```
If TypeName(n)= "Range" then n=n.Value
```

Effectuez ensuite la mise au point en pas à pas. Ce pas à pas démarrera dès que vous tenterez de réaliser un calcul grâce à votre fonction, soit en modifiant la valeur de la cellule source, soit en appuyant sur la touche **[F9] (Recalcul)**

Une erreur fréquente est l'oubli de la déclaration du paramètre formel **n** *dans l'en-tête de fonction* **`Function nromain(n).`**

Annexe : texte du module Module1

```
Function nromain(n)
    Dim j As Integer
    Dim k As Integer
    Dim r As String
    If TypeName(n) = "Range" Then n = n.Value
    If Not IsNumeric(n) Then
        nromain = CVErr(xlErrNum)
        Exit Function
    End If
    If n > 9999 Or n < 1 Then
        nromain = CVErr(xlErrValue)
        Exit Function
    End If
    r = ""
    'extraction milliers
    j = n \ 1000
    n = n - 1000 * j
    'conversion milliers
    For k = 1 To j
        r = r + "M"
    Next k
    'extraction centaines
    j = n \ 100
    n = n - 100 * j
    'conversion centaines
    Select Case j
        Case 1: r = r + "C"
        Case 2: r = r + "CC"
        Case 3: r = r + "CCC"
        Case 4: r = r + "CD"
        Case 5: r = r + "D"
        Case 6: r = r + "DC"
        Case 7: r = r + "DCC"
        Case 8: r = r + "DCCC"
        Case 9: r = r + "CM"
    End Select
    'extraction dizaines
    j = n \ 10
    n = n - 10 * j
    'conversion dizaines
    Select Case j
        Case 1: r = r + "X"
        Case 2: r = r + "XX"
        Case 3: r = r + "XXX"
        Case 4: r = r + "XL"
        Case 5: r = r + "L"
        Case 6: r = r + "LX"
        Case 7: r = r + "LXX"
        Case 8: r = r + "LXXX"
        Case 9: r = r + "XC"
    End Select
```

Suite du texte du module Module1

```
    'conversion unités
    Select Case n
        Case 1: r = r + "I"
        Case 2: r = r + "II"
        Case 3: r = r + "III"
        Case 4: r = r + "IV"
        Case 5: r = r + "V"
        Case 6: r = r + "VI"
        Case 7: r = r + "VII"
        Case 8: r = r + "VIII"
        Case 9: r = r + "IX"
    End Select
    nromain = r
End Function
```