PROGRAMMATION EXCEL EN VBA

Version 1.0.0
10 septembre 2006

Alain JAFFRE
(jack POINT r AT free POINT fr

Préface

Ce « livre » est tout particulièrement dédicacé à Isabelle qui a dû subir quelque chose qui ressemble plus à un bourrage de crâne qu'à un cours ou même une explication progressive de ce que c'est que la programmation et comment on peut la pratiquer avec Excel et VBA . Le manque de temps et la complexité du premier programme à réaliser sont en cause mais n'excuse pas tout. Qu'elle me pardonne.

L'écriture de ce document à donc été commencé suite à ce constat regrettable, afin que cela soit évité à d'autres.

La dernière version de ce document ainsi que les classeurs créés pour cette occasion sont téléchargeable sur http ://jack.r.free.fr

J'entends et j'oublie,
Je vois et je me souviens,
Je fais et je comprends.

Confucius

Remerciements

A mes parents qui m'ont inoculé le virus de l'apprentissage.

Si tu ne sais pas faire, sort les mains de tes poches et apprend !

A mes collègues de travail, qui m'obligent sans cesse à trouver de nouvelles solutions à leurs problèmes ...

A Anna, ma relectrice, qui a accepté d'être la candide qui me pointe du doigt les parties insuffisement claires.

A Daniel Flipo et Fabrice Popineau qui m'ont fait découvrir ce merveilleux outil qu'est LaTeX, lors d'une conférence des « Rencontres mondiales du logiciel libre » à Metz en 2003.

Aux participants du forum de discussion fr.comp.text.tex qui m'ont permis de progresser dans l'utilisation de LaTeX.

Licence

Voir le texte complet de la licence dans l'annexe A.

Conventions utilisées

Afin d'en faciliter la compréhension, ce document utilise les conventions suivantes :

- *Un texte comme celui-ci* est un texte que vous devez taper.
- Un texte comme celui-ci est un bouton que vous devez cliquer ou une touche sur laquelle vous devez appuyer.
- Un `mot` comme celui-ci est un mot clé (réservé) du language de programmation.
- 'Un texte comme celui-ci' est un texte que vous devez remplacer par la valeur correspondante.
- « Un texte comme celui-ci » est un paramètre d'une commande du langage. (Utilisé dans la référence du langage, annexe D)
- un exemple de syntaxe sera présenté comme cela :
  Syntaxe:

  ```
  syntaxe d'une commande
  ```

  Le code présenté n'est pas utilisable tel quel, il est juste là pour vous permettre de voir comment il sera écrit
- un exemple complet sera présenté comme cela :

Listing 1 – Exemple complet

```
Exemple complet de code.
Il vous est possible de le taper et de l'essayer.
```

Le code est utilisable tel quel. Voir page 2 pour la façon de le mettre en œuvre

- Un nombre entre crochets [] renvoi à la bibliographie, page 145

Révisions du document

Historique

- 1 octobre 2005 : Début d'écriture de ce document
- 1 novembre 2005 : Version 0.9 pour première relecture
- 20 avril 2006 : Version 0.9.1 avec référence du language
- 10 septembre 2006 : Version 1.0.0 relecture et première version publique

Table des matières

Chapitre 1

Introduction

1.1 Mise en garde

Je ne suis en aucun cas un programmeur professionnel. Tout ce que je « sais », je l'ai appris par essais successifs, échecs, réflexion et lecture de code écrit par d'autres. Mes façons de faire ne sont donc probablement pas très académiques mais, je l'espère, peuvent vous montrer que même si l'on ne sait pas faire, en prenant le temps et en étant motivé, tout le monde peut y arriver.

La version d'Excel utilisée lors de la rédaction de ce document est la version Excel 2002 SP3 française utilisée sur Windows XP Professionel SP1.

1.2 Qu'est-ce que la programmation dans Excel ?

Lorsque l'on effectue des manipulations répétitives dans Excel, il est interressant de faire en sorte de les automatiser. La première étape consiste à créer ces propre macros à l'aide de l'enregistreur de macro.

- On met en marche l'enregistreur → Outils , Macro , Nouvelle macro ,
- On effectue les manipulations,
- On arrête l'enregistreur → Outils , Macro , Arrêter l'enregistrement ,
- On peut maintenant ré-exécuter les mêmes manipulations en exécutant la macro enregistrée → Outils , Macro , Macros , choisir la macro voulue, Exécuter .

Sans le savoir, on vient de créer un programme dans Excel. L'enregistreur de macro a converti nos actions à la souris ou au clavier en un langage qui peut être relu par Excel. Le langage utilisé est le VBA.

1.3 Pourquoi programmer dans Excel en utilisant VBA ?

On a créé ces petites macros, mais rapidement, on se rend compte que si le nombre de lignes change, que la colonne a été déplacée, …, tout est à refaire. C'est là que va intervenir la programmation en VBA. En ayant un minimum de connaissances en VBA, on va pouvoir rendre générique quelque chose qui, à priori, ne l'était pas. On va apprendre au programme à trouver le nombre de lignes, à quelle place ce trouve la colonne qui nous interresse, …Nos petites macros vont ainsi pouvoir être

réutilisables, y compris dans d'autres classeurs. On pourra ainsi se constituer petit à petit une bibliothèque de macros et de fonctions que l'on aura testé, peaufinées et que l'on pourra réutiliser, sans pour autant réinventer la roue à chaque fois.

1.4 Comment voir, modifier le programme ?

Lorsque Excel est ouvert, on affiche l'éditeur VBA par ALT+F11 ou en passant par Outils , Macro , Visual Basic Editor . Affichez :

– l'explorateur de projet → Affichage , Explorateur de projets ,

– la fenêtre de propriété → Affichage , Fenêtre Propriétés .

Lorsque l'on **double clique** sur une feuille, ThisWorkbook ou un module, le texte du programme correspondant s'affiche dans la partie droite de l'écran. Dans la suite de ce document, nous continuerons à utiliser le mot programme pour parler du texte constituant nos macros et fonctions.

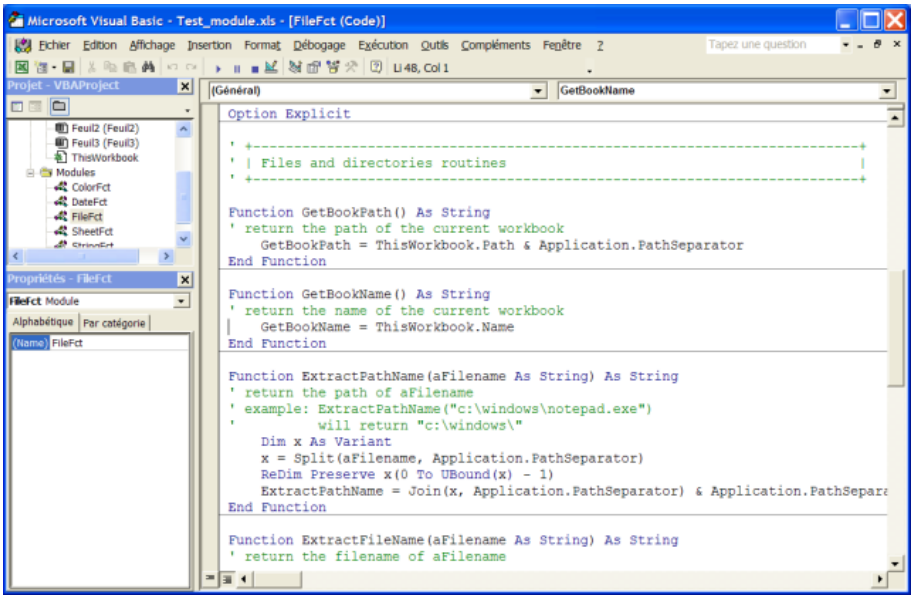

FIG. 1.1 – Editeur VBA

1.5 Comment enregistrer le programme ?

Lorsque vous êtes dans l'éditeur VBA, Fichier , Enregistrer . Vous pouvez aussi sauvegarder votre programme, simplement en enregistrant votre classeur.

1.6 Comment exécuter un programme (une macro) ?

Dans Excel, Outils , Macro , Macros , choisir la macro voulue, Exécuter . On verra par la suite comment ne rendre visible que les macros souhaitées (en fait en cachant les autres).

1.7 Comment importer, exporter, supprimer un module ?

1.7.1 Importer

Importer un module va permettre de rajouter des fonctions et macros déjà testées et fonctionnelles dans notre classeur. Elles seront ensuite utilisable comme n'importe qu'elle autres fonctions ou macros que l'on aurait écrite dans le classeur. Lorsque vous êtes dans l'éditeur VBA, faites un **clic droit** dans l'explorateur de projet, → Importer un fichier ... , choisir le fichier (module) à importer.

1.7.2 Exporter

Exporter un module va permettre d'enregistrer dans un fichier un groupe de fonctions et de macros que l'on a testé et qui peuvent nous être utiles dans d'autres projets. Lorsque vous êtes dans l'éditeur VBA, faites un **clic droit** sur le module que vous voulez exporter, → Exporter un fichier ... , donnez un nom au fichier qui contiendra votre module. Un certain nombre des modules que j'ai créé pour mes besoins personnels sont disponibles sur http ://jack.r.free.fr sous licence GPL (cf. annexe A).

1.7.3 Supprimer

Toutes les macros que vous ferez avec l'enregistreur de macro afin de voir comment programmer certaines actions seront placées dans des modules (module1, module2, ...). Lorsque celle-ci ne seront plus utiles, pensez à les supprimer du classeur, il s'allègera d'autant. Lorsque vous êtes dans l'éditeur VBA, faites un **clic droit** sur le module que vous voulez supprimer, → Supprimer nom_du_module , répondre à la question. Attention, il faut la lire avant de cliquer, elle ne correspond pas à ce que vous supposez !

1.8 Premier programme

En utilisant les explications ci-dessus, saisissez l'exemple ci-dessous dans ThisWorkbook puis exécutez-le.

Listing 1.1 – Premier programme

```
Option Explicit

Sub NumeroteLignes()
    Dim N As Long

    For N = 1 To 20
        Cells(N, 1) = "Ligne " & N
    Next N
End Sub
```

Vous venez de créer votre premier programme. si vous l'exécutez, vous verrez qu'il se contente d'afficher dans chaque cellule le numéro de ligne correspondant. Nous verrons au cours des explications qui suivent, à quoi correspondent chacun des termes utilisés.

1.9 Recommandations

1.9.1 Découpez

Avant de commencer un programme, prenez un papier, un crayon et listez toutes les étapes de ce que vous voulez réalisez. Détaillez au maximum afin que chaque tâche soit aussi simple que possible. Réalisez une à une les macros correspondant à ces tâches. Après chaque réalisation de macro, testez la et corrigez la. Vous verrez, il n'est pas rare que cela ne fonctionne pas du premier coup comme on l'espérait.

1.9.2 Enregistrez

Lorsque vous travaillez sur un programme, à chaque fois que vous avez écrit une portion qui fonctionne correctement, enregistrez votre travail sous un autre nom. Terminez le nom du fichier par un numéro que vous incrémentez par exemple. En procédant de la sorte, vous pourrez toujours revenir à une étape antérieure et serez assuré que le fonctionnement de votre programme n'est pas dépendant du nom du fichier qui le contient. C'est toujours très désagréable d'avoir un joli fichier qui ne fonctionne plus car un de vos collègue n'a rien trouvé de mieux que de le renommer, toujours pour une excellente raison !

1.9.3 Commentez

Commentez, commentez, commentez .... N'hésitez pas à mettre des commentaires qui vous permettront de retrouver à quoi sert telle ou telle portion du programme quelques temps plus tard. Pensez aussi aux personnes qui vous remplaceront un jour et qui auront besoin de découvrir comment cela fonctionne.

1.9.4 Indentez

Afin de permettre une meilleure lisibilité, il est préférable d'indenter le texte. Cela consiste à décaler à l'aide d'une tabulation les portions de codes représentant un sous-ensemble. Si on reprend notre premier programme, on se rend compte qu'avec une indentation correcte, on voit plus facilement que la ligne « Cells(N, 1) = "Ligne " & N » est répétée par la boucle `For`. Les deux exemples ci-dessous font la même chose mais l'un est plus facile à lire que l'autre.

Tout au long de ce document, les exemples de code seront indentés à ma façon. Inspirez-vous en.

Listing 1.2 – Sans indentation

```
Option Explicit

Sub NumeroteLignes()
Dim N As Long

For N = 1 To 20
Cells(N, 1) = "Ligne " & N
Next N
End Sub
```

Listing 1.3 – Avec indentation

```
Option Explicit

Sub NumeroteLignes()
    Dim N As Long

    For N = 1 To 20
        Cells(N, 1) = "Ligne " & N
    Next N
End Sub
```

1.9.5 Déclarez

Déclarez de façon explicite toutes vos variables. Cela clarifie le code du programme. En tête de chaque module , spécifiez `Option Explicit` afin d'obtenir un message d'erreur si vous tentez d'utiliser une variable non déclarée. Pour éviter de l'oublier, on peut configurer Excel pour que ce soit le fonctionnement par défaut. Dans l'éditeur VBA, Outils , Options , Editeur . Dans les paramètres du code, cochez Déclaration des variables obligatoire .

1.9.6 Structurez

Structurez votre programme. Définissez toujours une fonction ou une macro avant de l'utiliser, ainsi, lorsque vous la recherchez, vous savez qu'elle se trouve avant son utilisation. Cela vous évitera de chercher en aveugle où peut bien être cette ... de fonction ou macro lorsqu'il s'agira de la corriger ou de l'améliorer. Que ce soit dans un programme, un module, une fonction ou une macro, gardez toujours l'ordre déclaration (des variables, des fonctions, des macros) puis utilisation, quant bien même la déclaration dans le cours du programme fonctionne. Tout au long de ce document, les exemples de code respectent cette façon de faire. Inspirez-vous en.

	`Option Explicit`
	
	`Sub NumeroteLignes()`

`Déclaration`	`    Dim N As Long`

`Utilisation`	`    For N = 1 To 20`
	`        Cells(N, 1) = "Ligne " \& N`
	`    Next N`

	`End Sub`

TAB. 1.1 – Structure

Chapitre 2

Notions de base

2.1 Constantes et variables

Tout au long du programme, on va devoir stocker des informations que l'on voudra réutiliser plus tard. Cela se fera dans des constantes ou des variables.

2.1.1 Nom

Afin de pouvoir les identifier et réutiliser, chaque constante ou variable aura un nom. Celui-ci est constitué de lettres et chiffres mais ne contient pas d'espace. Pour éviter tout problème, n'utilisez pas de caractères exotiques, utilisez A..Z, a..z, 0..9 et _. Personnellement, je nomme tout en anglais, ce qui élimine le risque d'avoir mis un accent par inadvertance. Dans ce document, j'ai essayé de mettre des noms en français afin de les différencier plus facilement des mots utilisés par le langage. Les noms sont libres, sous réserve de ne pas utiliser un mot réservé du langage. Vous pouvez mélanger majuscules et minuscules de façon à faciliter leur lisibilité. Par exemple, CompteurLigne est plus lisible que compteurligne, même si d'un point de vue purement programmation cela ne change rien.

2.1.2 Constantes

Une constante est quelque chose qui est fixe et qui ne changera pas tout au long du programme. On va les utiliser pour des choses qui ne doivent pas évoluer au cours du programme mais que l'on souhaite pouvoir adapter facilement. Lors de leur définition, elles sont précédées du mot clé `Const`.

Exemple:

On va créer une feuille « *ma super feuille* » dans un classeur puis y faire appel plusieurs fois dans le programme. Si l'on écrit le nom de la feuille à chaque fois que l'on l'utilise dans le programme et que notre chef de service décide que son intitulé n'est pas bon, il va falloir tout relire et remplacer à chaque fois l'ancien nom par le nouveau nom. Si on a pris la précaution de définir le nom de la feuille dans une constante et d'utiliser cette constante tout au long du programme, il suffit de changer la valeur de la constante pour que tout le programme soit à jour.

Syntaxe:

```
Const TitrePage = "ma super feuille"
```

Le fait d'écrire `Const` suivi de son nom et de son contenu s'appel déclarer une constante.

2.1.3 Variables

Une variable est quelque chose qui va évoluer et changer de valeur tout au long du programme. On utilisera des variables pour ce qui change tel qu'un numéro de ligne ou colonne, le contenu d'une cellule que l'on veut modifier, ... Lors de leur définition, elles sont précédées du mot clé `Dim`, `Private` ou `Public` (c.f. Visibilité page 11).

Exemple:

On veut remplir des cellules qui se suivent avec un texte donné. Au lieu d'écrire autant de ligne que nécessaire, on va utiliser une variable pour effectuer une boucle qui se répétera le nombre de fois désiré.

Listing 2.1 – Variables

```
Option Explicit

Sub MaSuperMacro()
    Dim I As Integer

    For I = 1 to 100
        cells(1,I+3)= "mon super texte"
    Next I
End Sub
```

Le fait d'écrire `Dim` suivi d'un identifiant (nom) et d'un type s'appel déclarer une variable.

2.2 Commentaires

Très rapidement, le nombre de ligne de notre programme croît. Si l'on veut être capable de le relire et de comprendre ce qu'il fait deux ans plus tard, pour le modifier par exemple, il va être judicieux d'y mettre des commentaires. Il s'agit de texte libre où l'on peut mettre ce que l'on veut. Il est précédé d'une apostrophe.

Exemple:

On veut se souvenir du format à donner pour une date.

Syntaxe:

```
Dim DateDeDebut as String ' doit être sous la forme aaaammjj
```

2.3 Regroupement

2.3.1 Noms de macros, fonctions, modules

Les noms de macros, de fonctions ou de modules suivent les même régles que les constantes et variables (cf. 2.1.1 page 7).

2.3.2 Macros

Chaque macro réalise une suite d'opération. En exécutant une macro, ou en l'appelant dans un programme plus vaste, on va lui faire effectuer chacune des instructions qu'elle contient.

Exemple:
On veut ajuster les largeurs et hauteurs de toutes les cellules de la feuille.

Listing 2.2 – Macro

```
Option Explicit

Dim PosY As Long

Sub Ajuste(NomDeLaFeuille As String)
' ajuste la largeur et hauteur des cellules
    Sheets(NomDeLaFeuille).Select
    Cells.Select
    Cells.EntireColumn.AutoFit
    Cells.EntireRow.AutoFit
    Range("A1").Select
End Sub

Sub TstSub()
    ' Nomme la feuille
    ActiveSheet.Name = "Test"

    PosY = PosY + 1
    Cells(PosY, 1) = "Sub"
    PosY = PosY + 1
    Cells(PosY, 1) = "Call Ajuste(""Test"")"
    Call AutoAdjust("Test")"
    PosY = PosY + 1
End Sub
```

2.3.3 Fonctions

Une fonction réalise une suite d'opération et retourne une valeur.
Exemple:
On veut, à partir du nom complet d'un fichier (chemin + nom de fichier), récupérer le seul nom du fichier. On va donc écrire une macro qui fait cela. Elle commence par `Function` suivi d'un nom et, de façon facultative, de paramètres et se termine par `End Function`.

Listing 2.3 – Fonction

```
Option Explicit

Function ExtraitNomFichier(unFichier As String) As String
' retourne le nom de fichier contenu dans nomFichier
' exemple: ExtraitNomFichier("c:\windows\notepad.exe")
'          renverra "notepad.exe"
    Dim x As Variant
    x = Split(unFichier, Application.PathSeparator)
    ExtraitNomFichier = x(UBound(x))
End Function
```

```
Sub TstFonction
    Dim monFichier as String

    monFichier ="c:\mon\chemin\super\long\le_fichier.txt"
    Cells(1,1) = ExtraitNomFichier(monFichier)
End Sub
```

2.3.4 Modules

Rapidement, on va se construire une bibliothèque de macros et de fonctions. Il devient donc intéressant de les regrouper en fonction de leurs points commun (macros et fonctions qui agissent sur les feuilles de classeur, macros et fonctions qui travaillent sur les chaînes de caractères, ...). On va donc les rassembler au sein d'un module.

Exemple:

On peut créer le module FileFct qui contiendra toutes les fonctions que l'on va créer et qui auront un lien avec les fichiers.

Listing 2.4 – Module

```
' +-------------------------------------------------------+
' | Routines pour fichiers et dossiers                    |
' +-------------------------------------------------------+
Option Explicit

Function CheminClasseur() As String
' retourne le chemin du classeur courant
    CheminClasseur = ThisWorkbook.Path & Application.PathSeparator
End Function

Function NomClasseur() As String
' retourne le nom du classeur courant
    NomClasseur = ThisWorkbook.Name
End Function

Function ExtraitChemin(unFichier As String) As String
' retourne le chemin de unFichier
' exemple: ExtraitChemin("c:\windows\notepad.exe")
'          retournera "c:\windows\"
    Dim x As Variant
    x = Split(unFichier, Application.PathSeparator)
    ReDim Preserve x(0 To UBound(x) - 1)
    ExtraitChemin = Join(x, Application.PathSeparator)
    ExtraitChemin = ExtraitChemin  & Application.PathSeparator
End Function

Function ExtraitNom(unFichier As String) As String
' retourne le nom de unFichier
' exemple: ExtraitNom("c:\windows\notepad.exe")
```

```
'           retournera "notepad.exe"
    Dim x As Variant
    x = Split(unFichier, Application.PathSeparator)
    ExtraitNom = x(UBound(x))
End Function

Function FichierExiste(unFichier As String) As Boolean
' retourne true si le fichier existe
' exemple: FichierExiste("c:\windows\notepad.exe")
    FichierExiste = Dir(unFichier) <> ""
End Function

Function DossierExiste(unChemin As String) As Boolean
' retourne true si le dossier existe
' exemple: DossierExiste("c:\windows\")
    Dim x As String
    unChemin = ExtraitChemin(unChemin)
    On Error Resume Next
    x = GetAttr(unChemin) And 0
    If Err = 0 Then
        DossierExiste = True
    Else
        DossierExiste = False
    End If
End Function
```

2.4 Visibilité

Suivant la façon d'on nous avons déclaré une variable, une macro ou une fonction, elle sera pas utilisable partout. On parle aussi de portée des déclarations.

2.4.1 Structure d'un programme

Un programme est constitué de plusieurs éléments :
- ThisWorkBook, qui contient ce qui est propre au classeur,
- autant de modules que nécessaire qui contiennent les macros et fonctions dont nous avons besoin.

Chacuns de ces éléments est constitué d'un en-tête , qui contient les déclarations qui vont être valables pour tout l'élément, et d'un corps, qui contient les macros et fonctions.

2.4.2 Déclarations

Les déclarations de variables peuvent être implicites ou explicites.

Une déclaration implicite est réalisée lorsque l'on utilise une variable que l'on n'a pas déclarée précédement dans le programme. Excel défini alors lui-même le type de la variable le plus approprié. Cela semble facile mais se révéle très dangereux plus le programme devient important. Une faute de frappe et l'on a créé une nouvelle variable au lieu de réutiliser celle que l'on voulait.

Listing 2.5 – Déclaration implicite

```
Sub DeclarationImplicite
    For I = 1 to 100
        cells(1,I+3)= "mon super texte"
    Next I
End Sub
```

Une déclaration explicite est réalisée lorsque l'on va, avant utilisation, définir le type d'une variable.

Listing 2.6 – Déclaration explicite

```
Sub DeclarationExplicite
    Dim I As Integer

    For I = 1 to 100
        cells(1,I+3)= "mon super texte"
    Next I
End Sub
```

Afin de limiter le risque d'erreur, il est préférable de spécifier `Option Explicit` en en-tête de chaque module. Cela impose à Excel de vérifier que les variables ont été déclarées avant utilisation. Si jamais il trouve une variable non déclaré, il émet un message d'erreur. Pour éviter d'oublier de rajouter cette option, on peut configurer Excel pour qu'il soit systématiquement dans ce mode → Outils , Options , Editeur , cocher Déclaration de variable obligatoire

2.4.3 Portée

En utilisant les mots clés `Dim`, `Private`, `Static` et `Public`, on va indiquer qui pourra voir, utiliser la variable définie.

Une variable déclarée `Dim`, `Private` ou `Static` est une variable locale. Elle ne sera utilisable et connue que de la partie de programme (module, macro, fonction) qui la définie. `Private` n'est pas utilisable dans une macro (`sub`) ou une fonction (`function`). `Private` ne peut être utilisé qu'en entête de module.

Une variable déclarée `Dim` ou `Private` est réinitialisée à chaque lecture de sa déclaration. Une variable déclarée `Static` est initialisée seulement lors de la première lecture de sa déclaration. Par la suite, sa valeur ne fait qu'évoluer.

Une variable déclarée `Public` est une variable globale. Elle pourra être utilisée par n'importe qu'elle partie du programme. Afin d'éviter les risque de confusion, on déclarera de préférence les variables globales dans l'en-tête du module considéré.

Mise en évidence

Afin de mettre cela en évidence, j'ai crée le classeur « Test_Visibilite.xls ». Ce classeur, contient deux modules. Un certain nombre de variables sont définies et initialisées dans Module1. Ces variables sont ensuite affichées depuis les différents éléments du programme (macro principale, macro principale de Module1, boucle de répétition de Module 1, macro principale de Module2, boucle de répétition de Module2). On a bien évidement commenté l'option Explicit afin de ne pas générer d'erreur et de visualiser le contenu des variables non définies.

Code

Listing 2.7 – Portée - macro de test

```
' Code de ThisWorkBook:
' ********************

' Option Explicit

Sub Process()
    Cells.Select
    Selection.Clear
    Range("A1").Select

    Call ProcessModule1
    Call ProcessModule2

    Cells(1, 7) = "Valeurs dans Process"
    Cells(2, 7) = "Constante1 = "
    Cells(2, 8) = Constante1
    Cells(3, 7) = "VariablePublique1 = "
    Cells(3, 8) = VariablePublique1
    Cells(4, 7) = "VariablePrivate1 = "
    Cells(4, 8) = VariablePrivate1
    Cells(5, 7) = "VariableDim1 = "
    Cells(5, 8) = VariableDim1

    Cells(6, 7) = "Constante2 = "
    Cells(6, 8) = Constante2
    Cells(7, 7) = "VariableDim2 = "
    Cells(7, 8) = VariableDim2
    Cells(8, 7) = "VariableStatic2 = "
    Cells(8, 8) = VariableStatic2

End Sub
```

Listing 2.8 – Portée - module1

```
' Code de Module1:
' ***************

' Option Explicit

Const Constante1 = "Constante 1"
Public VariablePublic1 As String
Private VariablePrivate1 As String
Dim VariableDim1 As String

Sub RepeteModule1(Boucle As Integer)
    Dim Decalage As Integer

    Decalage = 10 * (Boucle - 1)

    Const Constante2 = "Constante 2"
    Dim VariableDim2 As String
    Static VariableStatic2 As Integer

    VariablePublic1 = "VariablePublic1 from Module1"
    VariablePrivate1 = "VariablePrivate1 from Module1"
    VariableDim1 = "VariableDim1 from Module1"

    VariableDim2 = "VariableDim2 from Module2"
    VariableStatic2 = VariableStatic2 + 1

    Cells(1 + Decalage, 1) = "Valeurs dans RepeteModule1"
    Cells(1 + Decalage, 2) = "Boucle " & Boucle
    Cells(2 + Decalage, 1) = "Constante1 = "
    Cells(2 + Decalage, 2) = Constante1
    Cells(3 + Decalage, 1) = "VariablePublic1 = "
    Cells(3 + Decalage, 2) = VariablePublic1
    Cells(4 + Decalage, 1) = "VariablePrivate1 = "
    Cells(4 + Decalage, 2) = VariablePrivate1
    Cells(5 + Decalage, 1) = "VariableDim1 = "
    Cells(5 + Decalage, 2) = VariableDim1

    Cells(6 + Decalage, 1) = "Constante2 = "
    Cells(6 + Decalage, 2) = Constante2
    Cells(7 + Decalage, 1) = "VariableDim2 = "
    Cells(7 + Decalage, 2) = VariableDim2
    Cells(8 + Decalage, 1) = "VariableStatic2 = "
    Cells(8 + Decalage, 2) = VariableStatic2

End Sub

Sub ProcessModule1()
    Dim I As Integer
```

```
    For I = 1 To 5
        RepeteModule1 (I)
    Next I

    Cells(1, 4) = "Valeurs dans ProcessModule1"
    Cells(2, 4) = "Constante1 = "
    Cells(2, 5) = Constante1
    Cells(3, 4) = "VariablePublic1 = "
    Cells(3, 5) = VariablePublic1
    Cells(4, 4) = "VariablePrivate1 = "
    Cells(4, 5) = VariablePrivate1
    Cells(5, 4) = "VariableDim1 = "
    Cells(5, 5) = VariableDim1

    Cells(6, 4) = "Constante2 = "
    Cells(6, 5) = Constante2
    Cells(7, 4) = "VariableDim2 = "
    Cells(7, 5) = VariableDim2
    Cells(8, 4) = "VariableStatic2 = "
    Cells(8, 5) = VariableStatic2
End Sub
```

Listing 2.9 – Portée - module2

```
' Code de Module2:
' ***************

' Option Explicit


Sub RepeteModule2(Boucle As Integer)
    Dim Decalage As Integer

    Decalage = 10 * (Boucle - 1)

    Cells(1 + Decalage, 11) = "Valeurs dans RepeteModule2"
    Cells(1 + Decalage, 12) = "Boucle " & Boucle
    Cells(2 + Decalage, 11) = "Constante1 = "
    Cells(2 + Decalage, 12) = Constante1
    Cells(3 + Decalage, 11) = "VariablePublic1 = "
    Cells(3 + Decalage, 12) = VariablePublic1
    Cells(4 + Decalage, 11) = "VariablePrivate1 = "
    Cells(4 + Decalage, 12) = VariablePrivate1
    Cells(5 + Decalage, 11) = "VariableDim1 = "
    Cells(5 + Decalage, 12) = VariableDim1

    Cells(6 + Decalage, 11) = "Constante2 = "
    Cells(6 + Decalage, 12) = Constante2
    Cells(7 + Decalage, 11) = "VariableDim2 = "
```

```
    Cells(7 + Decalage, 12) = VariableDim2
    Cells(8 + Decalage, 11) = "VariableStatic2 = "
    Cells(8 + Decalage, 12) = VariableStatic2

End Sub

Sub ProcessModule2()
    Dim I As Integer

    For I = 1 To 5
        RepeteModule2 (I)
    Next I

    Cells(1, 14) = "Valeurs dans ProcessModule2"
    Cells(2, 14) = "Constante1 = "
    Cells(2, 15) = Constante1
    Cells(3, 14) = "VariablePublic1 = "
    Cells(3, 15) = VariablePublic1
    Cells(4, 14) = "VariablePrivate1 = "
    Cells(4, 15) = VariablePrivate1
    Cells(5, 14) = "VariableDim1 = "
    Cells(5, 15) = VariableDim1

    Cells(6, 14) = "Constante2 = "
    Cells(6, 15) = Constante2
    Cells(7, 14) = "VariableDim2 = "
    Cells(7, 15) = VariableDim2
    Cells(8, 14) = "VariableStatic2 = "
    Cells(8, 15) = VariableStatic2
End Sub
```

Résultats

Valeurs dans Process	
Constante1 =	
VariablePublic1 =	VariablePublic1 from Module1
PrivateVar1 =	
VariableDim1 =	
Constante2 =	
VariableDim2 =	
VariableStatic2 =	

FIG. 2.1 – Visibilité dans la macro principale

Valeurs dans ProcessModule1	
Constante1 =	Constante 1
VariablePublic1 =	VariablePublic1 from Module1
PrivateVar1 =	PrivateVar1 from Module1
VariableDim1 =	VariableDim1 from Module1
Constante2 =	
VariableDim2 =	
VariableStatic2 =	

FIG. 2.2 – Visibilité dans la macro de module1

Valeurs dans ProcessModule2	
Constante1 =	
VariablePublic1 =	VariablePublic1 from Module1
PrivateVar1 =	
VariableDim1 =	
Constante2 =	
VariableDim2 =	
VariableStatic2 =	

FIG. 2.3 – Visibilité dans la macro de module2

Valeurs dans RepeteModule1	Boucle 1
Constante1 =	Constante 1
VariablePublic1 =	VariablePublic1 from Module1
PrivateVar1 =	PrivateVar1 from Module1
VariableDim1 =	VariableDim1 from Module1
Constante2 =	Constante 2
VariableDim2 =	VariableDim2 from Module2
VariableStatic2 =	1
Valeurs dans RepeteModule1	Boucle 2
Constante1 =	Constante 1
VariablePublic1 =	VariablePublic1 from Module1
PrivateVar1 =	PrivateVar1 from Module1
VariableDim1 =	VariableDim1 from Module1
Constante2 =	Constante 2
VariableDim2 =	VariableDim2 from Module2
VariableStatic2 =	2
Valeurs dans RepeteModule1	Boucle 3
Constante1 =	Constante 1
VariablePublic1 =	VariablePublic1 from Module1
PrivateVar1 =	PrivateVar1 from Module1
VariableDim1 =	VariableDim1 from Module1
Constante2 =	Constante 2
VariableDim2 =	VariableDim2 from Module2
VariableStatic2 =	3

FIG. 2.4 – Visibilité dans la répétition de module1

Valeurs dans RepeteModule2	Boucle 1
Constante1 =	
VariablePublic1 =	VariablePublic1 from Module1
PrivateVar1 =	
VariableDim1 =	
Constante2 =	
VariableDim2 =	
VariableStatic2 =	
Valeurs dans RepeteModule2	Boucle 2
Constante1 =	
VariablePublic1 =	VariablePublic1 from Module1
PrivateVar1 =	
VariableDim1 =	
Constante2 =	
VariableDim2 =	
VariableStatic2 =	

FIG. 2.5 – Visibilité dans la répétition de module2

Conclusion

- Seule la variable VariablePublic1 définie dans l'en-tête de Module1 est visible où que l'on soit dans le programme,
- Module2 ne voit que VariablePublic1. Tout le reste lui est inconnu,
- La macro principale de Module1 ne voit que les variables et constantes définies dans l'entête du module,
- La macro de répétition de Module1 elle, voit toutes les variables et constantes définies au sein d'elle même ou en en-tête de module,
- La variable VariableStatic2 s'initialise lors de la première exécution de la boucle puis garde sa valeur entre deux passages dans la boucle de répétition. ATTENTION, si l'on relance une nouvelle fois la macro principale Process, VariableStatic2 n'est pas remise à zéro, elle continue d'évoluer à chaque passage dans la boucle.

Chapitre 3

Types

3.1 Booléen

Il s'agit des valeurs vrai et faux. On les utilise beaucoup dans les tests. Lors de leur définition, ils sont déclarés par `Boolean`. Ils peuvent prendre les valeurs `False` (0) et `True` (toute valeur différente de 0)

Listing 3.1 – Booléen

```
Sub TstBooleen
    Const I = 20
    Const J = 30
    Dim Resultat As Boolean

    Resultat = I < J
    If Resultat
        cells(1,1)= "I est inférieur à J"
    Else
        cells(1,1)= "I est supérieur ou égal à J"
    End If
End Sub
```

3.2 Nombres

Catégorie	Mot réservé	Plage de valeurs
Octet	`Byte`	0 à 255
Entier	`Integer`	-32 768 à 32 767
Entier long	`Long`	-2 147 483 648 à 2 147 483 647
Réel simple	`Single`	-3,402823 E38 à 3,402823 E38
Réel double	`Double`	-1,79769313486231 E308 à 1,79769313486231 E308
Date	`Date`	1er janvier 100 au 31 décembre 9999
Monnaie	`Currency`	-922 337 203 685 477,5808 à 922 337 203 685 477,5807

TAB. 3.1 – Types de nombres

En cas de valeur dépassant la plage autorisée, une erreur de dépassement de capacité est générée.

On peut avoir jusqu'à 65535 lignes et colonnes dans Excel, il faut donc utiliser le type long pour tout ce qui à trait aux numéros de ligne ou colonne.

3.3 Chaînes de caractères

Lors de leur définition, elles sont déclarées par `String`.

Listing 3.2 – Chaînes de caractères

```
Sub TstChaines
    Dim UneChaine As String

    UneChaine = "Une chaine de caracteres"
    UneChaine = UneChaine & " que l'on vient de prolonger"
End Sub
```

On peut figer leur longeur en faisant suivre `String` d'une * et du nombre de caractères.

Syntaxe:

```
Dim UneChaineDe23Caracteres As String*23
```

3.4 Variant

Ce type de variable peut avoir un contenu de n'importe quel type (booléen, entier, réel, chaîne, ...). Il est à éviter autant que faire se peut car il consomme beaucoup de mémoire et ne permet pas de savoir le type de son contenu.

Listing 3.3 – Variant

```
Sub TstVariant
    Dim UnVariant As Variant

    UnVariant = "Une chaine de caracteres"
    UnVariant = True
    UnVariant = 3.141592654
End Sub
```

3.5 Définies par l'utilisateur

On peut aussi définir ces propres types.

Listing 3.4 – Défines par l'utilisateur

```
Sub TstUtilisateur
    Type TUtilisateur
        Nom As String
        Prenom As String
```

```
        Identifiant As String
        MotDePasse As String
    End Type

    Dim Client As TUtilisateur
    Dim Vendeur As TUtilisateur

    Client.Nom = "Doe"
    Client.Prenom = "John"
    Client.Identifiant = "jdoe"
    Client.MotDePasse = "inconnu"
End Sub
```

3.6 Conversions

Pour effectuer des conversions entre types, on utilise :

Type d'origine	Type final	Fonction	Remarques
Tous types	`Boolean`	CBool	Toute valeur non nulle est convertie en `True`, soit -1
Tous types	`Byte`	CByte	
Tous types	`Integer`	CInt	
Tous types	`Long`	CLng	
Tous types	`Single`	CSng	
Tous types	`Double`	CDbl	
Tous types	`Date`	CDate	Lors de la conversion d'une valeur numérique en date, la partie entiére correspond à la date, la partie réelle correspond à l'heure
Tous types	`Currency`	CCur	Nombre à virgule fixe. 15 chiffres avant la virgule, 4 après
Tous types	`String`	CStr	

TAB. 3.2 – Conversions de chaînes

3.7 Tests

Afin de vérifier les plages et fonctionnement de chaque types, j'ai réalisé le classeur « Test_Variable.xls ».

Chapitre 4

Structures de contrôles

4.1 Boucles

4.1.1 Do ... Loop Until

On utilisera une boucle `Do ...Loop Until` lorsque l'on veut que la boucle soit parcourue jusqu'à ce que la condition soit remplie.

Dans l'exemple ci-dessous, on exécute ce qui est entre `Do` et `Loop Until` jusqu'à ce que I devienne supérieur à 10.

Listing 4.1 – Do ... Loop Until

```
Sub TstDoLoopUntil
    Dim I As Long

    I = 1
    Do
        Cells(I + 1, 1) = I
        I = I + 1
    Loop Until I > 10
End Sub
```

affichera les valeurs 1 à 10.

Listing 4.2 – Do ... Loop Until - Attention

```
Sub TstDoLoopUntilAttention
    Dim I As Long

    I = 1
    Do
        Cells(I + 1, 2) = I
        I = I + 1
    Loop Until I > 0
End Sub
```

affichera 1. En effet, le test de condition n'est fait qu'en fin de boucle, elle est donc au moins parcourue une fois.

4.1.2 Do ... Loop While

On utilisera une boucle `Do ...Loop While` lorsque l'on veut que la boucle soit parcourue tant que la condition est remplie.

Dans l'exemple ci-dessous, on exécute ce qui est entre `Do` et `Loop While` tant que I est inférieur ou égal à 10.

Listing 4.3 – Do ... Loop While

```
Sub TstDoLoopWhile
    Dim I As Long

    I = 1
    Do
        Cells(I + 1, 4) = I
        I = I + 1
    Loop While I <= 10
End Sub
```

affichera les valeurs 1 à 10.

Listing 4.4 – Do ... Loop While - Attention

```
Sub TstDoLoopWhileAttention
    Dim I As Long

    I = 1
    Do
        Cells(I + 1, 4) = I
        I = I + 1
    Loop While I < 0
End Sub
```

affichera 1. En effet, le test de condition n'est fait qu'en fin de boucle, elle est donc au moins parcourue une fois, même si la condition n'est pas remplie au départ.

4.1.3 While ... Wend

Lorsque l'on voudra répéter une portion de programme tant qu'une condition est vrai, on utilisera une boucle `While ...Wend`.

Dans l'exemple ci-dessous, tant que I est inférieur ou égal à 10, on exécute ce qui se trouve entre `While` et `Wend`.

Listing 4.5 – While ... Wend

```
Sub TstWhileWend
    Dim I As Integer

    I = 1
    While I <= 10
        cells(I + 1, 4) = I
        I = I + 1
    Wend
End Sub
```

affichera les valeurs 1 à 10. La boucle ne sera parcourue que si la condition est remplie au départ. A opposer à la boucle `Do ...Loop While`.

4.1.4 For . . . Next

Lorsque l'on voudra répéter un nombre fini de fois une portion de programme, nous utiliserons une boucle `For ...Next`. La boucle sera parcourue le nombre de fois indiqué avec le pas specifié (écart entre chaque répétition) .

Listing 4.6 – For . . . Next - 1

```
Sub TstForNext1
    Dim I As Long

    For I = 1 to 10
        cells(1,I)= I
    Next I
End Sub
```

affichera les valeurs 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 dans des cellules successives.

Listing 4.7 – For . . . Next - 2

```
Sub TstForNext2
    Dim I As Long

    For I = 0 to 10 step 2
        cells(1,I/2)= I
    Next I
End Sub
```

affichera les valeurs 0, 2, 4, 6, 8, 10 dans des cellules successives.

Listing 4.8 – For . . . Next - 3

```
Sub TstForNext3
    Dim I As Long

    For I = 10 to 1 step -1
        cells(1,11 - I)= I
    Next I
End Sub
```

affichera les valeurs 10, 9, 8, 7, 6, 5, 4, 3, 2, 1 dans des cellules successives.

4.1.5 For Each ... Next

Lorsque l'on voudra parcourir un tableau, nous utiliserons une boucle `For Each ...Next`.

Listing 4.9 – For Each ... Next

```
Sub TstForEachNext
    Dim I As Long
    Dim Tableau(4) As String
    Dim Ville As Variant

    Tableau(0) = "Lille"
    Tableau(1) = "Roubaix"
    Tableau(2) = "Paris"
    Tableau(3) = "Lyon"
    Tableau(4) = "Marseille"

    Cells(13, 4) = "For Each Next"
    I = 1
    For Each Ville In Tableau
        Cells(I + 13, 4) = Ville
        I = I + 1
    Next
End Sub
```

La boucle est parcourue pour chacuns des éléments du tableau et affiche Lille, Roubaix, Paris, Lyon, Marseille.

4.1.6 Test

Les différents tests ont été réalisés dans le classeur « Test_Boucle.xls ».

4.2 Conditions

4.2.1 If ...Then ... Else ... End If

Pour un test simple, on utilisera `If ...Then ...End If`.

Listing 4.10 – If ...Then ... End If

```
Sub TstIfThenEndIf
    Dim I As Integer

    For I = 1 To 10
        Cells(I + 1, 1) = I
        If I = 5 Then
            Cells(I + 1, 2) = " I = 5"
        End If
    Next I
End Sub
```

affichera « I = 5 » lorsque I aura la valeur 5 et rien dans les autres cas.

Si la test à besoin d'être plus évolué, on utilisera la forme compléte `If ...Then ...Else...End If`.

Listing 4.11 – If...Then... Else... End If

```
Sub TstIfThenElseEndIf
    For I = 1 To 10
        Cells(I + 1, 4) = I
        If I < 5 Then
            Cells(I + 1, 5) = "I < 5"
        Else
            If I > 5 Then
                Cells(I + 1, 5) = "I > 5"
            Else
                Cells(I + 1, 5) = "I = 5"
            End If
        End If
    Next I
End Sub
```

affichera « I < 5 » lorsque I sera inférieur à 5 , « I = 5 » lorsque I aura la valeur 5, « I > 5 » lorsque I sera supérieur à 5 .

4.2.2 Select... Case

Lorsque l'on aura besoin de faire des tests multiples sur une variable, on utilisera `Select...Case`.

Listing 4.12 – Select... Case

```
Sub TstSelectCase
    Dim I As Integer

    For I = 1 To 10
        Cells(I + 13, 1) = I

        Select Case I
            Case 1
                Cells(I + 13, 2) = "I à la valeur 1"
            Case 2
                Cells(I + 13, 2) = "I à la valeur 2"
            Case 8 To 10
                Cells(I + 13, 2) = "I est compris entre 8 et 10"
            Case Else
                Cells(I + 13, 2) = "I est supérieur à 2"
        End Select
    Next I
End Sub
```

affichera « I à la valeur 1 » lorsque I sera égal à 1 , « I à la valeur 2 » lorsque I aura la valeur 2, « I est compris entre 8 et 10 » lorsque I aura les valeurs 8, 9 ou 10 et « I est supérieur à 2 » dans tous les autres cas.

4.2.3 Test

Les différents tests ont été réalisés dans le classeur « Test_Condition.xls ».

Annexe A

The GNU General Public License

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs ; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps : (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what

they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law : that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

   Activities other than copying, distribution and modification are not covered by this License ; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty ; keep intact all the notices that refer to this License and to the absence of any warranty ; and give any other recipients of the Program a copy of this License along with the Program.

   You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions :

   (a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

   (b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

   (c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including

an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception : if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you ; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following :

   (a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange ; or,

   (b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange ; or,

   (c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

   The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

   If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source

code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.
5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.
6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.
7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

   If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

   It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims ; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system ; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

   This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.
8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical

distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

   Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation ; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

Appendix : How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty ; and each file should have at least the “copyright” line and a pointer to where the full notice is found.

> one line to give the program's name and a brief idea of what it does.
> Copyright (C) yyyy name of author
>
> This program is free software ; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation ; either version 2 of the License, or (at your option) any later version.
>
> This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY ; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.
>
> You should have received a copy of the GNU General Public License along with this program ; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode :

> Gnomovision version 69, Copyright (C) yyyy name of author
> Gnomovision comes with ABSOLUTELY NO WARRANTY ; for details type ‘show w’.
> This is free software, and you are welcome to redistribute it under certain conditions ; type ‘show c’ for details.

The hypothetical commands `show w` and `show c` should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than `show w` and `show c` ; they could even be mouse-clicks or menu items—whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a “copyright disclaimer” for the program, if necessary. Here is a sample ; alter the names :

> Yoyodyne, Inc., hereby disclaims all copyright interest in the program ‘Gnomovision’ (which makes passes at compilers) written by James Hacker.
>
> signature of Ty Coon, 1 April 1989
> Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

Annexe B

Code / caractère ANSI

Code	Caractère
000	
001	
002	
003	
004	
005	|
006	
007	
008	
009	
010	
011	
012	
013	
014	
015	
016	
017	
018	
019	
020	
021	⊥
022	
023	⊣
024	
025	⊢
026	
027	
028	
029	
030	
031	
032	

Code	Caractère
033	!
034	”
035	#
036	$
037	%
038	&
039	’
040	(
041	)
042	*
043	+
044	,
045	-
046	.
047	/
048	0
049	1
050	2
051	3
052	4
053	5
054	6
055	7
056	8
057	9
058	:
059	;
060	<
061	=
062	>
063	?
064	@
065	A

Code	Caractère
066	B
067	C
068	D
069	E
070	F
071	G
072	H
073	I
074	J
075	K
076	L
077	M
078	N
079	O
080	P
081	Q
082	R
083	S
084	T
085	U
086	V
087	W
088	X
089	Y
090	Z
091	[
092	\
093	]
094	^
095	_
096	‘
097	a
098	b
099	c
100	d
101	e
102	f
103	g
104	h
105	i
106	j
107	k
108	l
109	m

Code	Caractère
110	n
111	o
112	p
113	q
114	r
115	s
116	t
117	u
118	v
119	w
120	x
121	y
122	z
123	{
124	|
125	}
126	~
127	
128	€
129	
130	,
131	
132	
133	…
134	†
135	‡
136	^
137	‰
138	Š
139	<
140	Œ
141	
142	Ž
143	
144	
145	‘
146	’
147	“
148	”
149	•
150	-
151	–
152	~
153	™

Code	Caractère
154	š
155	>
156	œ
157	
158	ž
159	Ÿ
160	
161	¡
162	¢
163	£
164	
165	¥
166	¦
167	§
168	¨
169	©
170	^{a}
171	«
172	
173	
174	®
175	
176	
177	±
178	
179	
180	’
181	μ
182	¶
183	·
184	¸
185	^{1}
186	^{0}
187	»
188	¼
189	½
190	¾
191	¿
192	À
193	Á
194	Â
195	Ã
196	Ä
197	Å

Code	Caractère
198	Æ
199	Ç
200	È
201	É
202	Ê
203	Ë
204	Ì
205	Í
206	Î
207	Ï
208	
209	Ñ
210	Ò
211	Ó
212	Ô
213	Õ
214	Ö
215	×
216	Ø
217	Ù
218	Ú
219	Û
220	Ü
221	Ý
222	
223	β
224	à
225	á
226	â
227	ã
228	ä
229	å
230	æ
231	ç
232	è
233	é
234	ê
235	ë
236	ì
237	í
238	î
239	ï
240	
241	ñ

Code	Caractère
242	ò
243	ó
244	ô
245	õ
246	ö
247	÷
248	ø
249	ù
250	ú
251	û
252	ü
253	ý
254	
255	ÿ

Annexe C

Table des caractères ANSI

Pour avoir la table complète avec la police et la configuration de votre ordinateur :

Listing C.1 – Table ANSI

```
Sub TableAnsi()
    Dim Ligne As Byte
    Dim Colonne As Byte
    Dim Valeur As Long

    ' Efface les cellules
    Cells.Select
    Selection.Clear
    ' Formatte les  cellules
    Selection.NumberFormat = "@"
    With Selection
        .HorizontalAlignment = xlCenter
        .VerticalAlignment = xlCenter
    End With
    Range("A1").Select

    ' Met le titre des colonnes
    For Colonne = 0 To 9
        Cells(1, Colonne + 2) = Colonne
    Next Colonne

    ' Met le titre des lignes
    For Ligne = 0 To 25
        Cells(Ligne + 2, 1) = Ligne & "_"
    Next Ligne

    ' Table
    For Ligne = 0 To 25
        For Colonne = 0 To 9
            Valeur = Ligne * 10 + Colonne
            If Valeur <= 255 Then
```

```
                Cells(Ligne + 2, Colonne + 2) = Chr(Valeur)
            End If
        Next Colonne
    Next Ligne

    ' Améliore le look
    Rows("1:1").Select
    Selection.Font.Bold = True
    Columns("A:A").Select
    Selection.Font.Bold = True
    Selection.HorizontalAlignment = xlRight
    Cells.Select
    Cells.EntireColumn.AutoFit
    Selection.ColumnWidth = 3
    Selection.RowHeight = 15
    Range("A1").Select
End Sub
```

Annexe D

Référence du langage

Dans la syntaxe, le termes entre [ ] sont facultatifs.

addition (+)

Type : Opérateur

Syntaxe : nombre + nombre

Description : Additionne deux nombres. Ces nombres peuvent être entier ou réel.

Exemples :
- 1 + 2 retourne 3
- 5.4 + 1.5 retourne 6.9
- −24.5 + (−2) retourne -26.5

Remarques :

Famille : Mathématique

Voir aussi : soustraction, multiplication, division, division entière, puissance

soustraction (-)

Type : Opérateur

Syntaxe : nombre - nombre

Description : Soustrait deux nombres. Ces nombres peuvent être entier ou réel.

Exemples :
- 1 − 2 retourne -1
- 5.4 − 1.5 retourne 3.9
- −24.5 − (−2) retourne 22.5

Remarques :

Famille : Mathématique

Voir aussi : addition, multiplication, division, division entière, puissance

multiplication (*)

Type : Opérateur

Syntaxe : nombre * nombre

Description : Multiplie deux nombres. Ces nombres peuvent être entier ou réel.

Exemples :
- $1 * 2$ retourne 2
- $5.4 * 1.5$ retourne 8.1
- $-24.5 * (-2)$ retourne 49

Remarques :

Famille : Mathématique

Voir aussi : addition, soustraction, division, division entière, puissance

division (/)

Type : Opérateur

Syntaxe : nombre / nombre

Description : Divise deux nombres. Ces nombres peuvent être entier ou réel.

Exemples :
- $1/2$ retourne 0.5
- $5.4/1.5$ retourne 3.6
- $-24.5/(-2)$ retourne 12.25

Remarques :

Famille : Mathématique

Voir aussi : addition, soustraction, multiplication, division entière, puissance

division entière (\)

Type : Opérateur

Syntaxe : nombre \ nombre

Description : Divise deux nombres. Ces nombres peuvent être entier ou réel. Le résultat est un entier. Si le nombre est réel, il est d'abord arrondi en un entier. A l'entier inférieur si la première décimale est inférieure à 5, à l'entier supérieur si la première décimale est égale ou supérieure à 5.

Exemples :
- $1\backslash 2$ retourne 0
- $5.4\backslash 1.5$ retourne 2 (interprété comme $5\backslash 2$)
- $-24.5\backslash(-2)$ retourne 12

Remarques :

Famille : Mathématique

Voir aussi : addition, division, multiplication, Mod, puissance, soustraction

puissance (^)

Type : Opérateur

Syntaxe : nombre ^exposant

Description : Élève le nombre fourni à la puissance spécifiée par l'exposant.

Exemples :
- 1 ^2 retourne 1
- 5.4 ^1.5 retourne 12.54846604
- -24.5 ^(-2) retourne -0.001665973

Remarques :

Famille : Mathématique

Voir aussi : addition, soustraction, multiplication, division, division entière, Sqr

concaténation (&)

Type : Opérateur

Syntaxe : texte & texte

Description : Met bout à bout deux chaînes de caractères. En cas de valeur numérique, elle est convertie en chaîne avant concaténation.

Exemples :
- "Mis bout " & "à bout" retourne Mis bout à bout
- 5 & 24.2 retourne 524.2

Remarques :

Famille : Chaînes de caractères

Voir aussi : Join

Abs()

Type : fonction

Syntaxe : abs(nombre)

Description : Retourne la valeur absolue du « nombre » que l'on fournit. Le « nombre » peut être un entier ou un réel.

Exemples :
- abs(2) retourne 2
- abs(-2) retourne 2
- abs(-2.4) retourne 2

Listing D.1 – Abs

```
Sub TstAbs()
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "Abs"
    PosY = PosY + 1
    Cells(PosY, 1) = "Abs(2)"
```

```
    Cells(PosY, 2) = Abs(2)
    PosY = PosY + 1
    Cells(PosY, 1) = "Abs(-2)"
    Cells(PosY, 2) = Abs(-2)
    PosY = PosY + 1
    Cells(PosY, 1) = "Abs(2.4)"
    Cells(PosY, 2) = Abs(2.4)
    PosY = PosY + 1
    Cells(PosY, 1) = "Abs(-2.4)"
    Cells(PosY, 2) = Abs(-2.4)
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Mathématique

Voir aussi : Fix, Int, Round

And

Type : Opérateur

Syntaxe :

1. condition1 And condition2
2. entier1 And entier2

Description :

1. Et logique, renvoi True (vrai) si les deux conditions sont remplies.
2. Et binaire, dans l'entier renvoyé, les bits à 1 sont ceux qui était à 1 aussi bien dans entier1 que dans entier2. Les autres sont à 0.

Exemples :

1. If (A = 5) And (B = 22) Then "Vrai"
2. 5 And 22 retourne 4 (00000101 And 00010110 donne 00000100)

Listing D.2 – And

```
Sub TstAnd()
    Dim A As Long
    Dim B As Long
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "And"
    A = 5
    B = 22
    PosY = PosY + 1
    Cells(PosY, 1) = "(A = 5) And (B = 22)"
    If (A = 5) And (B = 22) Then
        Cells(PosY, 2) = "Vrai"
    Else
```

```
        Cells(PosY, 2) = "Faux"
    End If
    PosY = PosY + 1
    Cells(PosY, 1) = "5 And 22"
    Cells(PosY, 2) = 5 And 22
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Opérateur logique

Voir aussi : Not, Or, Xor

Array

Type : fonction

Syntaxe : Array(valeur1, valeur2, valeur3, ..., valeurn)

Description : Retourne un tableau de valeurs. Les valeurs peuvent être, sauf indication contraire lors de la déclaration, de tous types et sont fournies dans une liste de valeurs séparées par des virgules.

Exemples : Avec Tableau = array(10,20,12,5,"Texte")
- Tableau(2) renvoi 12
- Tableau(4) renvoi Texte

Listing D.3 – Array

```
Sub TstArray()
    Dim Tableau
    Tableau = Array(10, 20, 12, 5, "Texte")
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "Array"
    PosY = PosY + 1
    Cells(PosY, 1) = "Tableau = Array(10,20,12,5,""Texte"")"
    PosY = PosY + 1
    Cells(PosY, 1) = "Tableau(2)"
    Cells(PosY, 2) = Tableau(2)
    PosY = PosY + 1
    Cells(PosY, 1) = "Tableau(4)"
    Cells(PosY, 2) = Tableau(4)
    PosY = PosY + 1
End Sub
```

Remarques : L'indice du tableau démarre à 0. Il faut donc faire Tableau(0) pour obtenir la première valeur.

Famille : Type de données

Voir aussi : ReDim

Asc

Type : fonction

Syntaxe : Asc(chaîne_de_caractères)

Description : Retourne le code ANSI du premier caractère de la « chaîne_de_caractères ».

Exemple : Asc("Test") retourne 84 qui est le code ANSI de la lettre T.

Listing D.4 – Asc

```
Sub TstAsc()
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "Asc"
    PosY = PosY + 1
    Cells(PosY, 1) = "Asc(""Test"")"
    Cells(PosY, 2) = Asc("Test")
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Chaîne de caractères

Voir aussi : Chr

Atn

Type : fonction

Syntaxe : Atn(nombre)

Description : Retourne l'arc tangente du « nombre » fournit. L'angle retourné est en radians.

Exemple : Atn(0.577) retourne 0.523336 (environ 30 degrés)

Listing D.5 – Atn

```
Sub TstAtn()
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "Atn"
    PosY = PosY + 1
    Cells(PosY, 1) = "Atn(0.577)"
    Cells(PosY, 2) = Atn(0.577)
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Mathématique

Voir aussi : Tan, Sin, Cos

Boolean

Type : Mot réservé

Syntaxe : variable As Boolean

Description : Défini la « variable » comme étant de type `booléan`, c'est à dire acceptant les valeurs vrai (True, toute valeur différente de 0) ou faux (False, 0).

Exemple : Dim resultat As Boolean

Remarques :

Famille : Programmation

Voir aussi : True, False

Byte

Type : Mot réservé

Syntaxe : variable As Byte

Description : Défini la « variable » comme étant de type `byte`, c'est à dire acceptant les valeurs allant de 0 à 255.

Exemple : Dim resultat As Byte

Remarques :

Famille : Programmation

Voir aussi : CByte

Call

Type : Mots réservés

Syntaxe : Call nom_de_macro

Description : Appel et exécute la macro spécifiée. Lorsque celle-ci est terminée, le programme continu à la ligne suivante, sauf s'il a été interrompu au sein de la macro.

Exemple : Call MaSuperMacro

Listing D.6 – Call

```
Dim PosY As Long

Private Sub Affiche
    Cells(PosY, 2) = "On est ici!"
End Sub

Sub TstCall()
    PosY = 1
    Cells(PosY, 1) = "Call"
    PosY = PosY + 1
    Cells(PosY, 1) = "Call Affiche"
    Call Affiche
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Programmation

Voir aussi : Sub

CBool

Type : Fonction

Syntaxe : CBool(valeur)

Description : Converti la « valeur » passée en en booléen. CBool retourne False (faux) uniquement si la valeur est le nombre 0 ou la chaîne de caractères "false".

Exemple :

- CBool(-5.3) retourne True
- CBool(-1) retourne True
- CBool(0) retourne False
- CBool(1) retourne True
- CBool("true") retourne True
- CBool("false") retourne False
- CBool("TRUE") retourne True

Listing D.7 – CBool

```
Sub TstCBool()
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "CBool"
    PosY = PosY + 1
    Cells(PosY, 1) = "CBool(-5.3)"
    Cells(PosY, 2) = CBool(-5.3)
    PosY = PosY + 1
    Cells(PosY, 1) = "CBool(-1)"
    Cells(PosY, 2) = CBool(-1)
    PosY = PosY + 1
    Cells(PosY, 1) = "CBool(0)"
    Cells(PosY, 2) = CBool(0)
    PosY = PosY + 1
    Cells(PosY, 1) = "CBool(1)"
    Cells(PosY, 2) = CBool(1)
    PosY = PosY + 1
    Cells(PosY, 1) = "CBool(5.3)"
    Cells(PosY, 2) = CBool(5.3)
    PosY = PosY + 1
    Cells(PosY, 1) = "CBool(""true"")"
    Cells(PosY, 2) = CBool("true")
    PosY = PosY + 1
    Cells(PosY, 1) = "CBool(""false"")"
    Cells(PosY, 2) = CBool("false")
    PosY = PosY + 1
    Cells(PosY, 1) = "CBool(""TRUE"")"
```

```
    Cells(PosY, 2) = CBool("TRUE")
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Conversion

Voir aussi : False, True

CByte

Type : Fonction

Syntaxe : CByte(valeur)

Description : Converti la « valeur » fournie en un nombre de type octet. Une erreur est déclenchée si la valeur n'est pas dans plage autorisé pour les `Byte` (0 à 255).

Exemple :

Listing D.8 – CByte

```
Sub TstCByte()
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "CByte"
    PosY = PosY + 1
    Cells(PosY, 1) = "CByte(0)"
    Cells(PosY, 2) = CByte(0)
    PosY = PosY + 1
    Cells(PosY, 1) = "CByte(255)"
    Cells(PosY, 2) = CByte(255)
    PosY = PosY + 1
    Cells(PosY, 1) = "CByte(""255"")"
    Cells(PosY, 2) = CByte("255")
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Conversion

Voir aussi : CInt, CLng, CSng, CDbl, CDate, CCur, CStr

CCur

Type : Fonction

Syntaxe : CCur(valeur)

Description : Converti la « valeur » fournie en un nombre de type monétaire. Une erreur est déclenchée si la valeur n'est pas dans la plage autorisé pour les `Currency` (-922 337 203 685 477,5808 à 922 337 203 685 477,5807).

Exemple : Sur un ordinateur configuré pour la monnaie Euro,
- CCur(-123456789.128) retourne -123 456 789.13 €
- CCur(-123456789.123) retourne -123 456 789.12 €
- CCur(123456789.123) retourne 123 456 789.12 €
- CCur(123456789.128) retourne 123 456 789.13 €

Listing D.9 – CCur

```
Sub TstCCur()
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "CCur"
    PosY = PosY + 1
    Cells(PosY, 1) = "CCur(-123456789.128)"
    Cells(PosY, 2) = CCur(-123456789.128)
    PosY = PosY + 1
    Cells(PosY, 1) = "CCur(-123456789.123)"
    Cells(PosY, 2) = CCur(-123456789.123)
    PosY = PosY + 1
    Cells(PosY, 1) = "CCur(123456789.123)"
    Cells(PosY, 2) = CCur(123456789.123)
    PosY = PosY + 1
    Cells(PosY, 1) = "CCur(123456789.128)"
    Cells(PosY, 2) = CCur(123456789.128)
    PosY = PosY + 1
End Sub
```

Remarques :
Famille : Conversion
Voir aussi : CByte, CInt, CLng, CSng, CDbl, CDate, CStr

CDate

Type : Fonction
Syntaxe : CDate(valeur)
Description : Converti la « valeur » fournie en une date.
Exemple :

Listing D.10 – CDate

```
Sub TstCDate()
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "CDate"
    PosY = PosY + 1
    Cells(PosY, 1) = "CDate(38723.35645)"
    Cells(PosY, 2) = CDate(38723.375)
    PosY = PosY + 1
End Sub
```

Remarques : La partie entière du nombre fourni correspond au nombre de jour depuis le 01/01/1900 tandis que la partie décimale correspond au nombre de secondes depuis 0h divisé par 86400.

Famille : Conversion

Voir aussi : CByte, CInt, CLng, CSng, CDbl, CCur, CStr

CDbl

Type : Fonction

Syntaxe : CDbl(valeur)

Description : Converti la « valeur » fournie en un nombre de type double. Une erreur est déclenchée si la valeur n'est pas dans la plage autorisé pour les `Double` (-1,79769313486231 E308 à 1,79769313486231 E308).

Exemple :

Listing D.11 – CDbl

```
Sub TstCDbl()
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "CDbl"
    PosY = PosY + 1
    Cells(PosY, 1) = "CDbl(""-1,79769313486231 E308"")"
    Cells(PosY, 2) = CDbl("-1,79769313486231 E308")
    PosY = PosY + 1
    Cells(PosY, 1) = "CDbl(""1,79769313486231 E308"")"
    Cells(PosY, 2) = CDbl("1,79769313486231 E308")
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Conversion

Voir aussi : CByte, CInt, CLng, CSng, CDate, CCur, CStr

Chr

Type : Fonction

Syntaxe : Chr(code_ANSI)

Description : Retourne le caractère correspondant au « code_ANSI » fournit.

Exemple : Chr(84) retourne la lettre T

Listing D.12 – Chr

```
Sub TstChr()
    Dim PosY As Long
```

```
    PosY = 1
    Cells(PosY, 1) = "Chr"
    PosY = PosY + 1
    Cells(PosY, 1) = "Chr(84)"
    Cells(PosY, 2) = Chr(84)
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Chaîne de caractères

Voir aussi : Asc

CInt

Type : Fonction

Syntaxe : CInt(valeur)

Description : Converti la « valeur » fournie en un nombre de type entier. Une erreur est déclenchée si la valeur n'est pas dans la plage autorisé pour les `Integer`(-32 768 à 32 767).

Exemple :

Listing D.13 – CInt

```
Sub TstCInt()
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "CInt"
    PosY = PosY + 1
    Cells(PosY, 1) = "CInt(-32768)"
    Cells(PosY, 2) = CInt(-32768)
    PosY = PosY + 1
    Cells(PosY, 1) = "CInt(32767)"
    Cells(PosY, 2) = CInt(32767)
    PosY = PosY + 1
    Cells(PosY, 1) = "CInt(""-32768"")"
    Cells(PosY, 2) = CInt("-32768")
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Conversion

Voir aussi : CByte, CLng, CSng, CDbl, CDate, CCur, CStr

CLng

Type : Fonction

Syntaxe : CLng(valeur)

Description : Converti la « valeur » fournie en un nombre de type entier long. Une erreur est déclenchée si la valeur n'est pas dans la plage autorisé pour les `Long` (-2 147 483 648 à 2 147 483 647).

Exemple :

Listing D.14 – CLng

```
Sub TstCLng()
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "CLng"
    PosY = PosY + 1
    Cells(PosY, 1) = "CLng(-2147483648)"
    Cells(PosY, 2) = CLng(-2147483648#)
    PosY = PosY + 1
    Cells(PosY, 1) = "CLng(2147483647)"
    Cells(PosY, 2) = CLng(2147483647#)
    PosY = PosY + 1
    Cells(PosY, 1) = "CLng(""-2 147 483 648"")"
    Cells(PosY, 2) = CLng("-2 147 483 648")
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Conversion

Voir aussi : CByte, CInt, CSng, CDbl, CDate, CCur, CStr

Const

Type : mot réservé

Syntaxe : Const nom = valeur

Description : Associe un « nom » avec une « valeur » qui ne changera pas tout au long du programme. En cas de modification, il suffit de changer la « valeur » pour que tout le programme soit mis à jour.

Exemple :

Listing D.15 – Const

```
Sub TstConst()
    Const titre = "Mon super titre"
    Const heureTravaille = 37.5
    Const tauxCredit = 3.65
    Dim PosY As Long
```

```
    PosY = 1
    Cells(PosY, 1) = "Const"
    PosY = PosY + 1
    Cells(PosY, 1) = "titre"
    Cells(PosY, 2) = titre
    PosY = PosY + 1
    Cells(PosY, 1) = "heureTravaille"
    Cells(PosY, 2) = heureTravaille
    PosY = PosY + 1
    Cells(PosY, 1) = "tauxCredit"
    Cells(PosY, 2) = tauxCredit
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Programmation

Voir aussi : Dim

Cos

Type : Fonction

Syntaxe : Cos(angle_en_radians)

Description : Retourne le cosinus de l'angle fournit. L'angle est en radians.

Exemple : Cos(0.523598) retourne 0.866025

Listing D.16 – Cos

```
Sub TstCos()
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "Cos"
    PosY = PosY + 1
    Cells(PosY, 1) = "Cos(0.523598)"
    Cells(PosY, 2) = Cos(0.523598)
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Mathématique

Voir aussi : Sin, Tan, Atn

CSng

Type : Fonction

Syntaxe : CSng(valeur)

Description : Converti la « valeur » fournie en un nombre de type réel simple. Une erreur est déclenchée si la « valeur » n'est pas dans la plage autorisé pour les `Single` (-3,402823 E38 à 3,402823 E38).

Exemple :

Listing D.17 – CSng

```
Sub TstCSng()
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "CSng"
    PosY = PosY + 1
    Cells(PosY, 1) = "CSng(""-3,402823 E38"")"
    Cells(PosY, 2) = CSng("-3,402823 E38")
    PosY = PosY + 1
    Cells(PosY, 1) = "CSng(""3,402823 E38"")"
    Cells(PosY, 2) = CSng("3,402823 E38")
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Conversion

Voir aussi : CByte, CInt, CLng, CDbl, CDate, CCur, CStr

CStr

Type : Fonction

Syntaxe : CStr(valeur)

Description : Converti la « valeur » fournie en une chaîne de caractères.

Exemple :

Listing D.18 – CStr

```
Sub TstCStr()
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "CStr"
    PosY = PosY + 1
    Cells(PosY, 1) = "CStr(-3.40)"
    Cells(PosY, 2) = CStr(-3.4)
    PosY = PosY + 1
    Cells(PosY, 1) = "CStr(23.45678)"
    Cells(PosY, 2) = "R=" & CStr(23.45678)
```

```
    ' "R=" est là pour forcer le mode texte dans la cellule
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Conversion

Voir aussi : CByte, CInt, CLng, CSng, CDbl, CDate, CCur

Currency

Type : Mot réservé

Syntaxe : variable As Currency

Description : Défini la « variable » comme étant de type `currency`, c'est à dire acceptant les valeurs allant de -922 337 203 685 477,5808 à 922 337 203 685 477,5807.

Exemple : Dim resultat As Currency

Remarques :

Famille : Programmation

Voir aussi : CCur

Date

Type : Fonction

Syntaxe : Date

Description : Retourne la date courante.

Exemple :

Listing D.19 – Date

```
Sub TstDate()
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "Date"
    PosY = PosY + 1
    Cells(PosY, 1) = "Date()"
    Cells(PosY, 2) = Date
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Date/Heure

Voir aussi : Now, Time

DateAdd

Type : Fonction

Syntaxe : DateAdd(unité,valeur_à_ajouter,date_origine)

Description : Ajoute « valeur_à_ajouter » à la « date_origine ». L'« unité » de l'ajout est définie par une chaîne de caractère qui peut prendre les valeurs suivantes :

Unité	Description
"yyyy"	année (year)
"q"	trimestre (quarter)
"m"	mois
"y"	jour de l'année (day of the year)
"d"	jour (day)
"w"	jour de la semaine (weekday)
"ww"	semaine (week)
"h"	heure
"n"	minute
"s"	seconde

TAB. D.1 – Valeurs pour unité

Exemple : DateAdd("q", 2, "01/01/2005") ajoute 2 trimestres au 1 janvier 2005 et retourne 01/07/2005.

Listing D.20 – DateAdd

```
Sub TstDateAdd()
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "DateAdd"
    PosY = PosY + 1
    Cells(PosY, 1) = "DateAdd(""yyyy"",1,""01/01/2005"")"
    Cells(PosY, 2) = DateAdd("yyyy", 1, "01/01/2005")
    PosY = PosY + 1
    Cells(PosY, 1) = "DateAdd(""m"",3,""01/01/2005"")"
    Cells(PosY, 2) = DateAdd("m", 3, "01/01/2005")
    PosY = PosY + 1
    Cells(PosY, 1) = "DateAdd(""d"",5,""01/01/2005"")"
    Cells(PosY, 2) = DateAdd("d", 5, "01/01/2005")
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Date/Heure

Voir aussi : DateDiff

DateDiff

Type : Fonction

Syntaxe : DateDiff(unité,première_date,deuxième_date [, premier_jour_de_la_semaine [, première_semaine_de_l_année]])

Description : Retourne l'écart qu'il y a entre deux dates.

- « Unité » est une chaîne de caractères qui indique le type d'écart que l'on veut connaître (voir tableau D.1 page 59)
- « premier_jour_de_la_semaine » est optionnel et indique quel est le premier jour de la semaine. S'il est omis, Excel prend Dimanche comme premier jour de la semaine. Il peut avoir les valeurs suivante :

Constante prédéfinie	Valeur	Description
vbUseSystem	0	utilise le réglage de l'API NLS
vbSunday	1	dimanche (valeur par défaut)
vbMonday	2	lundi
vbTuesday	3	mardi
vbWednesday	4	mercredi
vbThursday	5	jeudi
vbFriday	6	vendredi
vbSaturday	7	samedi

TAB. D.2 – Valeurs pour premier jour de la semaine

- « première_semaine_de_l_année » est optionnel et indique quel est la première semaine de l'année. S'il est omis, Excel prend la semaine qui contient le 1 janvier comme première semaine de l'année.

Constante prédéfinie	Valeur	Description
vbUseSystem	0	utilise le réglage de l'API NLS
vbFirstJan1	1	semaine qui contient le 1 janvier (valeur par défaut)
vbFirstFourDays	2	première semaine ayant au moins 4 jours
vbFirstFullWeek	3	première semaine complète de l'année

TAB. D.3 – Valeurs pour première semaine de l'année

Exemple :

Listing D.21 – DateDiff

```
Sub TstDateDiff()
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "DateDiff"
    PosY = PosY + 1
    Cells(PosY, 1) = "DateDiff(""yyyy"",""31/03/2002""," _
```

```
        & """01/01/2005"")"
    Cells(PosY, 2) = DateDiff("yyyy", "31/03/2002", _
        "01/01/2005")
    PosY = PosY + 1
    Cells(PosY, 1) = "DateDiff(""m"",""01/01/2005"",""  _
        & """15/05/2005"")"
    Cells(PosY, 2) = DateDiff("m", "01/01/2005", _
        "15/05/2005")
    PosY = PosY + 1
    Cells(PosY, 1) = "DateDiff(""d"",""01/01/2005"",""  _
        & """01/01/2006"")"
    Cells(PosY, 2) = DateDiff("d", "01/01/2005", _
        "01/01/2006")
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Date/Heure

Voir aussi : DateAdd

DatePart

Type : Fonction

Syntaxe : DatePart(unité, date [, premier_jour_de_la_semaine [, première_semaine_de_l_année]])

Description : Retourne la portion demandée de la date fournie.

- « Unité » défini la portion de date que l'on veut (voir tableau D.1 page 59).
- « premier_jour_de_la_semaine » est optionnel et indique quel est le premier jour de la semaine. S'il est omis, Excel prend Dimanche comme premier jour de la semaine (voir tableau D.2 page 60).
- « première_semaine_de_l_année » est optionnel et indique quel est la première semaine de l'année. S'il est omis, Excel prend la semaine qui contient le 1 janvier comme première semaine de l'année (voir tableau D.3 page 60).

Exemple :

Listing D.22 – DatePart

```
Sub TstDatePart()
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "DatePart"
    PosY = PosY + 1
    Cells(PosY, 1) = "DatePart(""yyyy"",""25/12/2005"")"
    Cells(PosY, 2) = DatePart("yyyy", "25/12/2005")
    PosY = PosY + 1
    Cells(PosY, 1) = "DatePart(""m"",""25/12/2005"")"
    Cells(PosY, 2) = DatePart("m", "25/12/2005")
    PosY = PosY + 1
```

```
    Cells(PosY, 1) = "DatePart(""d"",""25/12/2005"")"
    Cells(PosY, 2) = DatePart("d", "25/12/2005")
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Date/Heure

Voir aussi : Day, Month, Year

DateSerial

Type : Fonction

Syntaxe : DateSerial(année, mois, jour)

Description : Retourne la date correspondant au « jour », au « mois » et à l'« année » fournis.

Exemple :

Listing D.23 – DateSerial

```
Sub TstDateSerial()
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "DateSerial"
    PosY = PosY + 1
    Cells(PosY, 1) = "DateSerial(2005,12,25)"
    Cells(PosY, 2) = DateSerial(2005, 12, 25)
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Date/Heure

Voir aussi : DateValue

DateValue

Type : Fonction

Syntaxe : DateValue(chaîne_de_caractères)

Description : Retourne le numéro de série correspondant à la date fournie sous forme d'une chaîne de caractères.

Exemple :

Listing D.24 – DateValue

```
Sub TstDateValue()
    Dim PosY As Long
```

```
    PosY = 1
    Cells(PosY, 1) = "DateValue"
    PosY = PosY + 1
    Cells(PosY, 1) = "DateValue(""25/12/2005"")"
    Cells(PosY, 2) = DateValue("25/12/2005")
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Date/Heure

Voir aussi : DateSerial

Day

Type : Fonction

Syntaxe : Day(chaîne_de_caractères)

Description : Retourne le jour extrait de la date fournie sous forme d'une chaîne de caractères.

Exemple :

Listing D.25 – Day

```
Sub TstDay()
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "Day"
    PosY = PosY + 1
    Cells(PosY, 1) = "Day(""25/12/2005"")"
    Cells(PosY, 2) = Day("25/12/2005")
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Date/Heure

Voir aussi : DatePart, Month, Year

Dim

Type : Mot réservé

Syntaxe : Dim nom As type

Description : Défini une variable en lui fixant un nom et un type de donnée.

Exemple :

- Dim uneChaine As String
- Dim unEntier As Integer
- Dim unEntierLong As Long

- Dim unReel As Single
- Dim unReelDouble As Double

Remarques :

Famille : Programmation

Voir aussi : Const

Do ... Loop Until

Type : Mot réservé

Syntaxe : Do instructions Loop Until condition

Description : Effectue les « instructions » entre Do et Loop jusqu'à ce que la « condition » soit remplie.

Exemple :

Listing D.26 – Do ... Loop Until

```
Sub TstDoLoopUntil
    Dim I As Long

    I = 1
    Do
        Cells(I + 1, 1) = I
        I = I + 1
    Loop Until I > 10
End Sub
```

Remarques :

Famille : Programmation

Voir aussi : Do ... Loop While, For ... Next, For Each ... Next, While ... Wend

Do ... Loop While

Type : Mot réservé

Syntaxe : Do instructions Loop While condition

Description : Effectue les « instructions » entre Do et Loop tant que la « condition » est remplie. La « condition » est testée en fin de boucle.

Exemple :

Listing D.27 – Do ... Loop While

```
Sub TstDoLoopWhile
    Dim I As Long

    I = 1
    Do
        Cells(I + 1, 4) = I
        I = I + 1
    Loop While I <= 10
End Sub
```

Remarques :

Famille : Programmation

Voir aussi : Do … Loop Until, For … Next, For Each … Next, While … Wend

Double

Type : Mot réservé

Syntaxe : variable As Double

Description : Défini la « variable » comme étant de type `double`, c'est à dire acceptant les valeurs allant de -1,79769313486231 E308 à 1,79769313486231 E308.

Exemple : Dim resultat As Double

Remarques :

Famille : Programmation

Voir aussi : CDbl

Erase

Type : Mot réservé

Syntaxe : Erase tableau

Description : Vide les tableaux de taille prédéfinie ou réinitialise les tableaux de taille variable.

Exemple :

Listing D.28 – Erase

```
Sub TstErase()
    Dim Tableau(5)
    Tableau(1) = 10
    Tableau(2) = 1.5
    Tableau(5) = "Texte"

    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "Erase"
    PosY = PosY + 1
    Cells(PosY, 1) = "Tableau = Array(10,1.5,,""Texte"")"
    PosY = PosY + 1
    Cells(PosY, 1) = "Tableau(1)"
    Cells(PosY, 2) = Tableau(1)
    PosY = PosY + 1
    Cells(PosY, 1) = "Tableau(2)"
    Cells(PosY, 2) = Tableau(2)
    PosY = PosY + 1
    Cells(PosY, 1) = "Tableau(5)"
    Cells(PosY, 2) = Tableau(5)
```

```
    PosY = PosY + 1
    Cells(PosY, 1) = "Erase Tableau"
    Erase Tableau
    PosY = PosY + 1
    Cells(PosY, 1) = "Tableau(1)"
    Cells(PosY, 2) = Tableau(1)
    PosY = PosY + 1
    Cells(PosY, 1) = "Tableau(2)"
    Cells(PosY, 2) = Tableau(2)
    PosY = PosY + 1
    Cells(PosY, 1) = "Tableau(5)"
    Cells(PosY, 2) = Tableau(5)
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Programmation

Voir aussi :

Exit

Type : Mot réservé

Syntaxe :

1. Exit Do
2. Exit For
3. Exit Function
4. Exit Sub

Description : Sort de la boucle, de la macro ou de la fonction courante.

Exemple :

Listing D.29 – Exit

```
Sub TstExit()
    Dim N As Long
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "Exit"
    PosY = PosY + 1
    For N = 1 To 100
        Cells(PosY, 1) = "N vaut" & N
        If N = 5 Then Exit For
    Next N
    Cells(PosY, 1) = "On est sorti alors que N valait"
    Cells(PosY, 2) = N
    PosY = PosY + 1
    Do
```

```
        Cells(PosY, 1) = "N vaut" & N
        N = N + 1
        If N = 8 Then Exit Do
    Loop While N < 100
    Cells(PosY, 1) = "On est sorti alors que N valait"
    Cells(PosY, 2) = N
    PosY = PosY + 1
End Sub
```

Remarques :

Famille :

Voir aussi : Do ... Loop Until, Do ... Loop While, For ... Next, For Each ... Next, Function, Sub

Exp

Type : Fonction

Syntaxe : Exp(nombre)

Description : Retourne l'exponentielle de base e du « nombre » spécifié, avec e=2.71828183

Exemple :

Listing D.30 – Exp

```
Sub TstExp()
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "Exp"
    PosY = PosY + 1
    Cells(PosY, 1) = "Exp(3)"
    Cells(PosY, 2) = Exp(3)
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Mathématique

Voir aussi : Log

False

Type : Mot réservé

Syntaxe : False

Description : Type booléen valant faux (0). On l'utilise beaucoup dans tout ce qui est test, comparaison.

Exemple :

Listing D.31 – False

```
Sub TstFalse()
    Dim Trouve As Boolean

    Dim PosY As Long

    PosY = 1
    Trouve = False
    Cells(PosY, 1) = "False"
    PosY = PosY + 1
    Cells(PosY, 1) = "if Trouve = False then" _
        & " Cells(PosY,2) = ""Perdu"""
    If Trouve = False Then
        Cells(PosY, 2) = "Perdu"
    Else
        Cells(PosY, 2) = "Trouvé"
    End If
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Type de données

Voir aussi : True

Filter

Type : Fonction

Syntaxe : Filter(tableau_à_une_dimension, chaîne_à_rechercher, contenant_la_chaîne)

Description : Retourne un tableau contenant les chaînes du « tableau_à_une_dimension » de départ qui contiennent (« contenant_la_chaîne » = True) ou non (« contenant_la_chaîne » = False) la « chaîne_à_rechercher ».

Exemple :

Listing D.32 – Filter

```
Sub TstFilter()
    Dim resultat As Variant
    Dim text As String
    Dim n As Long
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "Filter"
    PosY = PosY + 1
    Cells(PosY, 1) = "Filter(Array(""accelere""," _
        & " ""freine"", ""camion"", ""excel""," _
```

```
        & " ""elephant""), ""el"", True)"
    resultat = Filter(Array("accelere", "freine", _
        "camion", "excel", "elephant"), "el", True)
    text = ""
    For n = LBound(resultat) To UBound(resultat)
        text = text & resultat(n) & ", "
    Next n
    Cells(PosY, 2) = text
    PosY = PosY + 1
    Cells(PosY, 1) = "Filter(Array(""accelere"",_
        & " ""freine"", ""camion"", ""excel"","_
        & " ""elephant""), ""el"", False)"
    resultat = Filter(Array("accelere", "freine", _
        "camion", "excel", "elephant"), "el", False)
    text = ""
    For n = LBound(resultat) To UBound(resultat)
        text = text & resultat(n) & ", "
    Next n
    Cells(PosY, 2) = text
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Chaînes de caractères

Voir aussi :

Fix

Type : Fonction

Syntaxe : Fix(réel)

Description : Retourne la partie entière d'un « réel ».

Exemple :

Listing D.33 – Fix

```
Sub TstFix()
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "Fix"
    PosY = PosY + 1
    Cells(PosY, 1) = "Fix(-2.16)"
    Cells(PosY, 2) = Fix(-2.16)
    PosY = PosY + 1
    Cells(PosY, 1) = "Fix(2.16)"
    Cells(PosY, 2) = Fix(2.16)
    PosY = PosY + 1
    Cells(PosY, 1) = "Fix(2.96)"
    Cells(PosY, 2) = Fix(2.96)
```

```
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Mathématique

Voir aussi : Abs, Int, Round

For . . . Next

Type : Mot réservé

Syntaxe : For variable=plage instructions Next variable

Description : Exécute les « instructions » entre For et Next le nombre de fois spécifié par la « variable » de comptage. La « plage » de valeurs que va prendre la variable de comptage est définie en début de boucle et ne peut être modifiée en cours de route.

Exemple :

Listing D.34 – For . . . Next

```
Sub TstForNext
    Dim I As Long

    For I = 1 to 10
        cells(1,I)= I
    Next I
End Sub
```

Remarques :

Famille : Programmation

Voir aussi : Do . . . Loop Until, Do . . . Loop While, For Each . . . Next, While . . . Wend

For Each . . . Next

Type : Mot réservé

Syntaxe : For Each variable In tableau instructions Next

Description : Exécute les instructions entre For Each et Next. On parcourt tous les éléments d'un tableau en stockant la valeur courante dans la « variable » fournie.

Exemple :

Listing D.35 – For Each . . . Next

```
Sub TstForEachNext
    Dim I As Long
    Dim Tableau(4) As String
    Dim Ville As Variant
```

```
    Tableau(0) = "Lille"
    Tableau(1) = "Roubaix"
    Tableau(2) = "Paris"
    Tableau(3) = "Lyon"
    Tableau(4) = "Marseille"

    Cells(13, 4) = "For Each Next"
    I = 1
    For Each Ville In Tableau
        Cells(I + 13, 4) = Ville
        I = I + 1
    Next
End Sub
```

Remarques :

Famille : Programmation

Voir aussi : Do ... Loop Until, Do ... Loop While, For ... Next, While ... Wend

FormatCurrency

Type : Fonction

Syntaxe : FormatCurrency(valeur [, nombre_de_décimales [, zéro_de_tête [, parenthèses_pour_les_nombres_négatifs [, groupe_les_chiffres]]]])

Description : Formate la valeur fournie comme une monnaie.

- « nombre_de_décimales » indique le nombre de décimales voulues. S'il est omis, on utilise les paramètres régionaux de l'ordinateur.
- « zéro_de_tête » indique si l'on affiche les zéro de tête (vbFalse, vbTrue, vbUseDefault). S'il est omis ou mis à vbUseDefault, on utilise les paramètres régionaux de l'ordinateur.
- « parenthèses_pour_les_nombres_négatifs » indique si l'on met les chiffres négatifs entre parenthèses (vbFalse, vbTrue, vbUseDefault). S'il est omis ou mis à vbUseDefault, on utilise les paramètres régionaux de l'ordinateur.
- « groupe_les_chiffres » indique si l'on regroupe les chiffres par 3 pour améliorer la lisibilité (vbFalse, vbTrue, vbUseDefault). S'il est omis ou mis à vbUseDefault, on utilise les paramètres régionaux de l'ordinateur.

Exemple :

Listing D.36 – FormatCurrency

```
Sub TstFormatCurrency()
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "FormatCurrency"
    PosY = PosY + 1
    Cells(PosY, 1) = "FormatCurrency(""1234567,123"", 2)"
    Cells(PosY, 2) = FormatCurrency("1234567,123", 2)
    PosY = PosY + 1
```

```
    Cells(PosY, 1) = "FormatCurrency(""-1234567,123"", 2," _
        & " vbTrue, vbTrue, vbFalse)"
    Cells(PosY, 2) = FormatCurrency("-1234567,123", 2, _
        vbTrue, vbTrue, vbFalse)
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Formatage

Voir aussi : FormatDateTime, FormatNumber, FormatPercent

FormatDateTime

Type : Fonction

Syntaxe : FormatDateTime(valeur [, type_de_format])

Description : Formate la « valeur » fournie comme une information de date et d'heure. « type_de_format » peut avoir les valeurs suivantes :

Constante prédéfinie	Valeur	Description
vbGeneralDate	0	Affiche une date et/ou une heure. S'il y a une partie date, elle est affichée au format court. S'il y a une partie heure, elle est affichée au format long. S'il y a une partie date et une partie heure, les deux sont affichées
vbLongDate	1	Affiche la date au format long défini dans les paramètres régionaux de l'ordinateur
vbShortDate	2	Affiche la date au format court défini dans les paramètres régionaux de l'ordinateur
vbLongTime	3	Affiche l'heure au format défini dans les paramètres régionaux de l'ordinateur
vbShortTime	4	Affiche l'heure sur 24 heures (hh :mm)

TAB. D.4 – Valeurs pour les formats des dates

Exemple :

Listing D.37 – FormatDateTime

```
Sub TstFormatDateTime()
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "FormatDateTime"
```

```
    PosY = PosY + 1
    Cells(PosY, 1) = _
    "FormatDateTime(""26/12/2005 10:44:21"","" _
        & " vbShortTime)"
    Cells(PosY, 2) = FormatDateTime("26/12/2005 10:44:21", _
        vbShortTime)
    PosY = PosY + 1
    Cells(PosY, 1) = _
    "FormatDateTime(""26/12/2005 10:44:21"","" _
        & " vbLongTime)"
    Cells(PosY, 2) = FormatDateTime("26/12/2005 10:44:21", _
        vbLongTime)
    PosY = PosY + 1
    Cells(PosY, 1) = _
    "FormatDateTime(""26/12/2005 10:44:21"","" _
        & " vbShortDate)"
    Cells(PosY, 2) = FormatDateTime("26/12/2005 10:44:21", _
        vbShortDate)
    PosY = PosY + 1
    Cells(PosY, 1) = _
    "FormatDateTime(""26/12/2005 10:44:21"","" _
        & " vbLongDate)"
    Cells(PosY, 2) = FormatDateTime("26/12/2005 10:44:21", _
        vbLongDate)
    PosY = PosY + 1
    Cells(PosY, 1) = _
    "FormatDateTime(""26/12/2005 10:44:21"","" _
        & " vbGeneralDate)"
    Cells(PosY, 2) = FormatDateTime("26/12/2005 10:44:21", _
        vbGeneralDate)
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Formatage

Voir aussi : FormatCurrency, FormatNumber, FormatPercent

FormatNumber

Type : Fonction

Syntaxe : FormatNumber(valeur [, nombre_de_décimales [, zéro_de_tête [, parenthèses_pour_les_nombres_négatifs [, groupe_les_chiffres]]]])

Description : Formate un nombre.

- « nombre_de_décimales » indique le nombre de décimales voulues. S'il est omis, on utilise les paramètres régionaux de l'ordinateur.
- « zéro_de_tête » indique si l'on affiche les zéro de tête (vbFalse, vbTrue, vbUseDefault). S'il est omis ou mis à vbUseDefault, on utilise les paramètres régionaux de l'ordinateur.

– « parenthèses_pour_les_nombres_négatifs » indique si l'on met les chiffres négatifs entre parenthèses (vbFalse, vbTrue, vbUseDefault). S'il est omis ou mis à vbUseDefault, on utilise les paramètres régionaux de l'ordinateur.
– « groupe_les_chiffres » indique si l'on regroupe les chiffres par 3 pour améliorer la lisibilité (vbFalse, vbTrue, vbUseDefault). S'il est omis ou mis à vbUseDefault, on utilise les paramètres régionaux de l'ordinateur.

Exemple :

Listing D.38 – FormatNumber

```
Sub TstFormatNumber()
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "FormatNumber"
    PosY = PosY + 1
    Cells(PosY, 1) = "FormatNumber(""1234567,1234"",3)"
    Cells(PosY, 2) = FormatNumber("1234567,1234", 3)
    PosY = PosY + 1
    Cells(PosY, 1) = "FormatNumber(""1234567,1234"", 3," _
        & " vbFalse, vbFalse, vbTrue)"
    Cells(PosY, 2) = FormatNumber("1234567,1234", 3, _
        vbFalse, vbFalse, vbTrue)
    PosY = PosY + 1
    Cells(PosY, 1) = "FormatNumber(""-1234567,1234"", 3," _
        & " vbTrue, vbTrue, vbTrue)"
    Cells(PosY, 2) = FormatNumber("-1234567,1234", 3, _
        vbTrue, vbTrue, vbTrue)
    PosY = PosY + 1
    Cells(PosY, 1) = "FormatNumber(""0,123456"", 3," _
        & " vbFalse, vbFalse, vbTrue)"
    Cells(PosY, 2) = FormatNumber("0,123456", 3, _
        vbFalse, vbFalse, vbTrue)
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Formatage

Voir aussi : FormatCurrency, FormatDateTime, FormatPercent

FormatPercent

Type : Fonction

Syntaxe : FormatPercent(valeur [, nombre_de_décimales [, zéro_de_tête [, parenthèses_pour_les_nombres_négatifs [, groupe_les_chiffres]]]])

Description : Formate un pourcentage.

– « nombre_de_décimales » indique le nombre de décimales voulues. S'il est omis, on utilise les paramètres régionaux de l'ordinateur.

– « zéro_de_tête » indique si l'on affiche les zéro de tête (vbFalse, vbTrue, vbUseDefault). S'il est omis ou mis à vbUseDefault, on utilise les paramètres régionaux de l'ordinateur.
– « parenthèses_pour_les_nombres_négatifs » indique si l'on met les chiffres négatifs entre parenthèses (vbFalse, vbTrue, vbUseDefault). S'il est omis ou mis à vbUseDefault, on utilise les paramètres régionaux de l'ordinateur.
– « groupe_les_chiffres » indique si l'on regroupe les chiffres par 3 pour améliorer la lisibilité (vbFalse, vbTrue, vbUseDefault). S'il est omis ou mis à vbUseDefault, on utilise les paramètres régionaux de l'ordinateur.

Exemple :

Listing D.39 – FormatPercent

```
Sub TstFormatPercent()
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "FormatPercent"
    PosY = PosY + 1
    Cells(PosY, 1) = "FormatPercent(""123,1234"",3)"
    Cells(PosY, 2) = FormatPercent("123,1234", 3)
    PosY = PosY + 1
    Cells(PosY, 1) = "FormatPercent(""0,00123456"", 2," _
        & " vbTrue, vbFalse, vbTrue)"
    Cells(PosY, 2) = FormatPercent("0,00123456", 2, _
    vbTrue, vbFalse, vbTrue)
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Formatage

Voir aussi : FormatCurrency, FormatDateTime, FormatNumber

Function

Type : Mot réservé

Syntaxe : Function Nom_de_la_fonction ... End Function

Description : Réalise la suite d'opération qu'elle contient et peut retourner une valeur.

Exemple :

```
Dim PosY As Long

Function ExtraitNomFichier(unFichier As String) As String
' retourne le nom de fichier contenu dans unFichier
' exemple: ExtraitNomFichier("c:\windows\notepad.exe")
'          renverra "notepad.exe"
    Dim x As Variant
    x = Split(unFichier, Application.PathSeparator)
```

```
    ExtraitNomFichier = x(UBound(x))
End Function

Sub TstFonction
    Dim monFichier as String

    monFichier ="c:\mon\chemin\super\long\le_fichier.txt"
    Cells(1,1) = ExtraitNomFichier(monFichier)
End Sub
```

Remarques :

Famille : Programmation

Voir aussi : Call, Sub

Hex

Type : Fonction

Syntaxe : Hex(entier)

Description : Retourne la valeur hexadécimale correspondant à l'« entier » fourni.

Exemple :

Listing D.40 – Hex

```
Sub TstHex()
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "Hex"
    PosY = PosY + 1
    Cells(PosY, 1) = "Hex(255)"
    Cells(PosY, 2) = Hex(255)
    PosY = PosY + 1
    Cells(PosY, 1) = "Hex(2598)"
    Cells(PosY, 2) = Hex(2598)
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Conversion

Voir aussi : Oct

Hour

Type : Fonction

Syntaxe : Hour(chaîne_de_caractères)

Description : Retourne l'heure extraite de la « chaîne_de_caractères » fournie qui représente des heures, minutes, secondes.

Exemple :

Listing D.41 – Hour

```
Sub TstHour()
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "Hour"
    PosY = PosY + 1
    Cells(PosY, 1) = "Hour(""10:23:52"")"
    Cells(PosY, 2) = Hour("10:23:52")
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Date/Heure

Voir aussi : Minute, Second

If . . . Then . . . Else

Type : Mot réservé

Syntaxe : If condition Then instructions_si_vrai Else instructions_si_faux End If

Description : Effectue un test sur la « condition » fournie. Si elle est vrai alors les « instructions_si_vrai » à la suite de Then sont exécutées, sinon on exécute les « instructions_si_faux » à la suite du Else, s'il existe. End If marque la fin du bloc d'instructions réalisées ou non en fonction du test.

Exemple :

```
Sub TstIfThenEndIf
    Dim I As Integer

    For I = 1 To 10
        Cells(I + 1, 1) = I
        If I = 5 Then
            Cells(I + 1, 2) = " I = 5"
        End If
    Next I
End Sub
```

Remarques :

Famille : Programmation

Voir aussi : Select . . . Case

InStr

Type : Fonction

Syntaxe : InStr(début, chaîne_de_caractères, texte_recherché)

Description : Retourne la première position du « texte_recherché » dans la « chaîne_de_caractères » en commençant la recherche à la position « début » de cette même chaîne.

Exemple :

Listing D.42 – InStr

```
Sub TstInStr()
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "InStr"
    PosY = PosY + 1
    Cells(PosY, 1) = "InStr(3, " & _
        """anticonstitutionnellement""" & _
        ", ""ti"")"
    Cells(PosY, 2) = InStr(3, _
        "anticonstitutionnellement", _
        "ti")
    PosY = PosY + 1
    Cells(PosY, 1) = "InStr(4, " & _
        """anticonstitutionnellement""" & _
        ", ""ti"")"
    Cells(PosY, 2) = InStr(4, _
        "anticonstitutionnellement", _
        "ti")
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Chaînes de caractères

Voir aussi : InStrRev, strcomp

InStrRev

Type : Fonction

Syntaxe : InStrRev(fin, chaîne_de_caractères, texte_recherché)

Description : Retourne la dernière position du « texte_recherché » dans la « chaîne_de_caractères » en terminant la recherche à position « fin » de cette même chaîne.

Exemple :

Listing D.43 – InStrRev

```
Sub TstInStrRev()
    Dim PosY As Long
```

```
    PosY = 1
    Cells(PosY, 1) = "InStrRev"
    PosY = PosY + 1
    Cells(PosY, 1) = "InStrRev(" & _
        """anticonstitutionnellement""" & _
        ", ""ti"")"
    Cells(PosY, 2) = InStrRev( _
        "anticonstitutionnellement", _
        "ti")
    PosY = PosY + 1
    Cells(PosY, 1) = "InStrRev("  & _
        """anticonstitutionnellement""" & _
        ", ""ti"",12)"
    Cells(PosY, 2) = InStrRev( _
        "anticonstitutionnellement", _
        "ti", 12)
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Chaînes de caractères

Voir aussi : InStr, strcomp

Int

Type : Fonction

Syntaxe : Int(réel)

Description : Retourne la partie entière d'un « réel » en l'arrondissant à l'entier immédiatement inférieur.

Exemple :

Listing D.44 – Int

```
Sub TstInt()
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "Int"
    PosY = PosY + 1
    Cells(PosY, 1) = "Int(123.456)"
    Cells(PosY, 2) = Int(123.456)
    PosY = PosY + 1
    Cells(PosY, 1) = "Int(123.987)"
    Cells(PosY, 2) = Int(123.987)
    PosY = PosY + 1
    Cells(PosY, 1) = "Int(-123.456)"
    Cells(PosY, 2) = Int(-123.456)
    PosY = PosY + 1
    Cells(PosY, 1) = "Int(-123.987)"
```

```
    Cells(PosY, 2) = Int(-123.987)
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Mathématique

Voir aussi : Abs, Fix, Round

Integer

Type : Mot réservé

Syntaxe : variable As Integer

Description : Défini la « variable » comme étant de type `integer`, c'est à dire acceptant les valeurs allant de -32 768 à 32 767.

Exemple : Dim resultat As Integer

Remarques :

Famille : Programmation

Voir aussi : CInt, Fix, Int

IsArray

Type : Fonction

Syntaxe : IsArray(variable)

Description : Retourne `True` si « variable » est une array, `False` dans le cas contraire.

Exemple :

Listing D.45 – IsArray

```
Sub TstIsArray()
    Dim maVariable As Variant
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "IsArray"
    maVariable = Array("bleu", "blanc", "rouge")
    PosY = PosY + 1
    Cells(PosY, 1) = "maVariable = " & _
        "Array(""bleu"", ""blanc"", ""rouge"")" & _
        Chr(10) & "IsArray(maVariable)"
    Cells(PosY, 2) = IsArray(maVariable)

    maVariable = "test"
    PosY = PosY + 1
    Cells(PosY, 1) = "maVariable = ""test""" & _
        Chr(10) & "IsArray(maVariable)"
    Cells(PosY, 2) = IsArray(maVariable)
```

```
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Type de données

Voir aussi : IsDate, IsEmpty, IsNull, IsNumeric, IsObject

IsDate

Type : Fonction

Syntaxe : IsDate(variable)

Description : Retourne `True` si « variable » est une date, `False` dans le cas contraire.

Exemple :

Listing D.46 – IsDate

```
Sub TstIsDate()
    Dim maVariable As Variant
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "IsDate"
    maVariable = Date
    PosY = PosY + 1
    Cells(PosY, 1) = "maVariable = date" & _
        Chr(10) & "IsArray(maVariable)"
    Cells(PosY, 2) = IsDate(maVariable)

    maVariable = "test"
    PosY = PosY + 1
    Cells(PosY, 1) = "maVariable = ""test""" & _
        Chr(10) & "IsDate(maVariable)"
    Cells(PosY, 2) = IsDate(maVariable)
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Type de données

Voir aussi : IsArray, IsEmpty, IsNull, IsNumeric, IsObject

IsEmpty

Type : Fonction

Syntaxe : IsEmpty(variable)

Description : Retourne `True` si « variable » a été initialisée, `False` dans le cas contraire.

Exemple :

Listing D.47 – IsEmpty

```
Sub TstIsEmpty()
    Dim maVariable As Variant
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "IsEmpty"
    PosY = PosY + 1
    Cells(PosY, 1) = "maVariable non initialisée" & _
        Chr(10) & "IsEmpty(maVariable)"
    Cells(PosY, 2) = IsEmpty(maVariable)

    maVariable = "test"
    PosY = PosY + 1
    Cells(PosY, 1) = "maVariable = ""test""" & _
        Chr(10) & "IsEmpty(maVariable)"
    Cells(PosY, 2) = IsEmpty(maVariable)
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Type de données

Voir aussi : IsArray, IsDate, IsNull, IsNumeric, IsObject

IsNull

Type : Fonction

Syntaxe : IsNull(variable)

Description : Retourne `True` si « variable » ne contient aucune donnée valide, `False` dans le cas contraire.

Exemple :

Listing D.48 – IsNull

```
Sub TstIsNull()
    Dim maVariable As Variant
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "IsNull"
    maVariable = Null
    PosY = PosY + 1
    Cells(PosY, 1) = "maVariable = null" & _
        Chr(10) & "IsNull(maVariable)"
    Cells(PosY, 2) = IsNull(maVariable)
```

```
    maVariable = "test"
    PosY = PosY + 1
    Cells(PosY, 1) = "maVariable = ""test""" & _
        Chr(10) & "IsNull(maVariable)"
    Cells(PosY, 2) = IsNull(maVariable)
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Type de données

Voir aussi : IsArray, IsDate, IsEmpty, IsNumeric, IsObject

IsNumeric

Type : Fonction

Syntaxe : IsNumeric(variable)

Description : Retourne `True` si « variable » contient une valeur numérique, `False` dans le cas contraire.

Exemple :

Listing D.49 – IsNumeric

```
Sub TstIsNumeric()
    Dim maVariable As Variant
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "IsNumeric"
    maVariable = 123
    PosY = PosY + 1
    Cells(PosY, 1) = "maVariable = 123" & _
        Chr(10) & "IsNumeric(maVariable)"
    Cells(PosY, 2) = IsNumeric(maVariable)

    maVariable = -123
    PosY = PosY + 1
    Cells(PosY, 1) = "maVariable = -123" & _
        Chr(10) & "IsNumeric(maVariable)"
    Cells(PosY, 2) = IsNumeric(maVariable)

    maVariable = 123.456
    PosY = PosY + 1
    Cells(PosY, 1) = "maVariable = 123.456" & _
        Chr(10) & "IsNumeric(maVariable)"
    Cells(PosY, 2) = IsNumeric(maVariable)

    maVariable = -123.456
    PosY = PosY + 1
    Cells(PosY, 1) = "maVariable = -123.456" & _
```

```
        Chr(10) & "IsNumeric(maVariable)"
    Cells(PosY, 2) = IsNumeric(maVariable)

    maVariable = "123"
    PosY = PosY + 1
    Cells(PosY, 1) = "maVariable = ""123""" & _
        Chr(10) & "IsNumeric(maVariable)"
    Cells(PosY, 2) = IsNumeric(maVariable)

    maVariable = "test"
    PosY = PosY + 1
    Cells(PosY, 1) = "maVariable = ""test""" & _
        Chr(10) & "IsNumeric(maVariable)"
    Cells(PosY, 2) = IsNumeric(maVariable)
    PosY = PosY + 1

End Sub
```

Remarques :

Famille : Type de données

Voir aussi : IsArray, IsDate, IsEmpty, IsNull, IsObject

IsObject

Type : Fonction

Syntaxe : IsObject(variable)

Description : Retourne `True` si « variable » est un `object`, `False` dans le cas contraire.

Exemple :

Listing D.50 – IsObject

```
Sub TstIsObject()
    Dim maVariable As Variant

    PosY = PosY + 1
    Cells(PosY, 1) = "IsObject"
    Set maVariable = CreateObject("Excel.Sheet")
    PosY = PosY + 1
    Cells(PosY, 1) = "maVariable = " & _
        "CreateObject(""Excel.Sheet"")" & _
        Chr(10) & "IsObject(maVariable)"
    Cells(PosY, 2) = IsObject(maVariable)

    maVariable = "test"
    PosY = PosY + 1
    Cells(PosY, 1) = "maVariable = ""test""" & _
        Chr(10) & "IsObject(maVariable)"
    Cells(PosY, 2) = IsObject(maVariable)
```

```
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Type de données

Voir aussi : IsArray, IsDate, IsEmpty, IsNull, IsNumeric

Join

Type : Fonction

Syntaxe : Join(tableau, chaîne_de_séparation)

Description : Retourne l'ensemble des éléments du « tableau » mis bout à bout en les séparant par la « chaîne_de_séparation ».

Exemple :

Listing D.51 – Join

```
Sub TstJoin()
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "Join"
    PosY = PosY + 1
    Cells(PosY, 1) = "Join(Array(""Aimer"", " & _
        """apprendre"", ""lire""), "", "")"
    Cells(PosY, 2) = Join(Array("Aimer", _
        "apprendre", "lire"), ", ")
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Chaînes de caractères

Voir aussi : concaténation

LBound

Type : Fonction

Syntaxe : LBound(tableau)

Description : Retourne le plus petit indice du « tableau ».

Exemple :

Listing D.52 – LBound

```
Sub TstLBound()
    Dim liste As Variant
    Dim PosY As Long
```

```
    liste = Array("Aimer", "apprendre", "lire")
    PosY = 1
    Cells(PosY, 1) = "LBound"
    PosY = PosY + 1
    Cells(PosY, 1) = "LBound(liste) -> "  & _
        "liste(LBound(liste))" & Chr(10) & _
        "avec liste = Array(""Aimer"", " & _
        """apprendre"", ""lire"")"
    Cells(PosY, 2) = LBound(liste) & " -> " & _
    liste(LBound(liste))
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Tableau

Voir aussi : UBound

LCase

Type : Fonction

Syntaxe : LCase(chaîne_de_caractères)

Description : Retourne la « chaîne_de_caractères » fournie, convertie en minuscules.

Exemple :

Listing D.53 – LCase

```
Sub TstLCase()
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "LCase"
    PosY = PosY + 1
    Cells(PosY, 1) = "LCase(""TexTe A MetTrE EN MInusCule"")"
    Cells(PosY, 2) = LCase("TexTe A MetTrE EN MInusCule")
    PosY = PosY + 1
End Sub
```

Remarques :

Famille :

Voir aussi : UCase

Left

Type : Fonction

Syntaxe : Left(chaîne_de_caractères, nombre)

Description : Retourne « nombre » de caractères indiqué qui correspondent à la partie droite de la « chaîne_de_caractères » fournie.

Exemple :

Listing D.54 – Left

```
Sub TstLeft()
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "Left"
    PosY = PosY + 1
    Cells(PosY, 1) = "Left(""Programmer en VBA"", 5)"
    Cells(PosY, 2) = Left("Programmer en VBA", 5)
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Chaînes de caractères

Voir aussi : Len, Mid, Right

Len

Type : Fonction

Syntaxe : Len(chaîne_de_caractères)

Description : Retourne le longueur de la « chaîne_de_caractères ».

Exemple :

Listing D.55 – Len

```
Sub TstLen()
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "Len"
    PosY = PosY + 1
    Cells(PosY, 1) = "Len(""Programmer en VBA"")"
    Cells(PosY, 2) = Len("Programmer en VBA")
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Chaînes de caractères

Voir aussi : Left, Right

Log

Type : Fonction

Syntaxe : Log(nombre_positif_ou_nul)

Description : Retourne le logarithme népérien du « nombre_positif_ou_nul » fourni.

Exemple :

Listing D.56 – Log

```
Sub TstLog()
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "Log"
    PosY = PosY + 1
    Cells(PosY, 1) = "Log(10)"
    Cells(PosY, 2) = Log(10)
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Mathématique

Voir aussi : Exp

Long

Type : Mot réservé

Syntaxe : variable As Long

Description : Défini la « variable » comme étant de type `long`, c'est à dire acceptant les valeurs allant de -2 147 483 648 à 2 147 483 647.

Exemple : Dim resultat As Long

Remarques :

Famille : Programmation

Voir aussi : CLng,

LTrim

Type : Fonction

Syntaxe : LTrim(chaîne_de_caractères)

Description : Retourne la « chaîne_de_caractères » après en avoir supprimé les espaces se trouvant à gauche.

Exemple :

Listing D.57 – LTrim

```
Sub TstLTrim()
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "LTrim"
    PosY = PosY + 1
    Cells(PosY, 1) = "->LTrim(""    Programmer en VBA    "")<-"
```

```
    Cells(PosY, 2) = "->" & LTrim("    Programmer en VBA    ") _
    & "<-"
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Chaînes de caractères

Voir aussi : RTrim, Trim

Mid

Type : Fonction

Syntaxe : Mid(chaîne_de_caractères, début, longueur)

Description : Retourne la portion de la « chaîne_de_caractères » qui commence à « début » et de « longueur » caractères.

Exemple :

Listing D.58 – Mid

```
Sub TstMid()
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "Mid"
    PosY = PosY + 1
    Cells(PosY, 1) = "Mid(""Programmer en VBA"", 5, 4)"
    Cells(PosY, 2) = Mid("Programmer en VBA", 5, 4)
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Chaînes de caractères

Voir aussi : Left, Right

Minute

Type : Fonction

Syntaxe : Minute(chaîne_de_caractères)

Description : Retourne les minutes extraite de la « chaîne_de_caractères » fournie, qui représente des heures, minutes, secondes.

Exemple :

Listing D.59 – Minute

```
Sub TstMinute()
    Dim PosY As Long
```

```
    PosY = 1
    Cells(PosY, 1) = "Minute"
    PosY = PosY + 1
    Cells(PosY, 1) = "Minute(""10:23:52"")"
    Cells(PosY, 2) = Minute("10:23:52")
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Date/Heure

Voir aussi : Hour, Second

Mod

Type : Opérateur

Syntaxe : entier1 Mod entier2

Description : Retourne le reste de la division de « entier1 » par « entier2 ». Il s'agit du modulo.

Exemple :

Listing D.60 – Mod

```
Sub TstMod()
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "Mod"
    PosY = PosY + 1
    Cells(PosY, 1) = "9 Mod 2"
    Cells(PosY, 2) = 9 Mod 2
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Mathématique

Voir aussi : division entière

Month

Type : Fonction

Syntaxe : Month(chaîne_de_caractères)

Description : Retourne le mois extrait de la date fournie sous forme d'une « chaîne_de_caractères ».

Exemple :

Listing D.61 – Month

```
Sub TstMonth()
```

```
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "Month"
    PosY = PosY + 1
    Cells(PosY, 1) = "Month(""25/12/2005"")"
    Cells(PosY, 2) = Month("25/12/2005")
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Date/Heure

Voir aussi : DatePart, Day, Year

MonthName

Type : Fonction

Syntaxe : MonthName(mois [,court])

Description : Retourne le nom du « mois » fourni. Si « court » vaut True, alors on obtient la forme abrégée du nom du mois.

Exemple :

Listing D.62 – MonthName

```
Sub TstMonthName()
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "MonthName"
    PosY = PosY + 1
    Cells(PosY, 1) = "MonthName(Month(""25/12/2005""))"
    Cells(PosY, 2) = MonthName(Month("25/12/2005"))
    PosY = PosY + 1
    Cells(PosY, 1) = "MonthName(Month(""25/12/2005""), True)"
    Cells(PosY, 2) = MonthName(Month("25/12/2005"), True)
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Date/Heure

Voir aussi : WeekdayName

Not

Type : Opérateur

Syntaxe : Not condition

Description : Négation de ce qui suit. Très utilisé dans les tests conditionnels.

Exemple :

- if not (A = B) then ...(si a n'est pas égal à B)
- if (A And (not B)) then ...(si A est vrai mais pas B)

Remarques :

Famille : Opérateur logique

Voir aussi : And, Or, Xor

Now

Type : Fonction

Syntaxe : Now()

Description : Retourne la date et l'heure courante.

Exemple :

Listing D.63 – Now

```
Sub TstNow()
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "Now"
    PosY = PosY + 1
    Cells(PosY, 1) = "Now()"
    Cells(PosY, 2) = Now()
    Dim PosY As Long

    PosY = 1
End Sub
```

Remarques :

Famille : Date/Heure

Voir aussi : Date

Nothing

Type : Mots réservés

Syntaxe : Nothing

Description : Supprime l'assignation d'un objet.

Exemple : Set mon_objet = Nothing

Remarques :

Famille : Programmation

Voir aussi :

Null

Type : Mots réservés

Syntaxe : Null

Description : Supprime l'affectation d'une variable. Elle ne contient alors plus aucune donnée valide.

Exemple : mavariable = Null

Remarques :

Famille : Programmation

Voir aussi :

Oct

Type : Fonction

Syntaxe : Oct(entier)

Description : Retourne la valeur octale (base 8) correspondant à l'« entier » fourni.

Exemple :

Listing D.64 – Oct

```
Sub TstOct()
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "Oct"
    PosY = PosY + 1
    Cells(PosY, 1) = "Oct(255)"
    Cells(PosY, 2) = Oct(255)
    PosY = PosY + 1
    Cells(PosY, 1) = "Oct(2598)"
    Cells(PosY, 2) = Oct(2598)
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Conversion

Voir aussi : Hex

Or

Type : Opérateur

Syntaxe :

1. condition1 Or condition2
2. entier1 Or entier2

Description :

1. Ou logique, renvoi True (vrai) si l'une des deux ou les deux conditions sont remplies.
2. Ou binaire, dans l'entier renvoyé, les bits à 1 sont ceux qui était à 1 dans « entier1 » et / ou dans « entier2 ». Les autres sont à 0.

Exemples :

1. If (A = 5) Or (B = 22) Then "Vrai"
2. 5 Or 22 retourne 23 (00000101 And 00010110 donne 00010111)

Listing D.65 – Or

```
Sub TstOr()
    Dim A As Long
    Dim B As Long

    PosY = PosY + 1
    Cells(PosY, 1) = "Or"
    A = 5
    B = 22
    PosY = PosY + 1
    Cells(PosY, 1) = "(A = 5) Or (B = 22)"
    If (A = 5) Or (B = 22) Then
        Cells(PosY, 2) = "Vrai"
    Else
        Cells(PosY, 2) = "Faux"
    End If
    PosY = PosY + 1
    Cells(PosY, 1) = "5 Or 22"
    Cells(PosY, 2) = 5 Or 22
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Opérateur logique

Voir aussi : And, Not, Xor

Private

Type : Mots réservés

Syntaxe : Private définition_de_macro_ou_de_variable

Description : Rend disponible la macro ou la variable, dont la définition suit, uniquement dans le module courant. Elle ne pourra pas être utilisée dans un autre module ou ThisWorkbook. Cela permet de la protéger et d'éviter des problèmes de nommage (nom de macro ou de fonction identiques utilisés dans différents modules).

Exemple :

Listing D.66 – Private

```
' Code de ThisWorkbook:
' ***********************

Option Explicit

Sub process()

    posY = 1
    Call ExecModule1
    Call ExecModule2
End Sub

' Code de TstPublicPrivateModule1:
' ********************************

Option Explicit

Public posY As Long
Private Valeur As Single

Private Sub Affiche()
    Cells(posY, 1) = "Je suis dans la macro ""Affiche"" du'' _
        & "module 1."
    Cells(posY, 2) = "Je suis en ligne " & posY _
        & " grâce à ""posY"" qui est public."
    Cells(posY, 3) = "La ""Valeur"" private est: " & Valeur
    posY = posY + 1
End Sub

Public Sub DisBonjour(nbDeFois As Long)
    Dim n As Long
    For n = 1 To nbDeFois
        Cells(posY, 1) = "Bonjour depuis le module 1."
        posY = posY + 1
    Next n
End Sub

Sub ExecModule1(Optional dummy As Boolean)
' dummy ne sert qu'a empêcher que je n'apparaisse
' dans le liste des macros du menu outils, macros

    Valeur = 1.23456
    Call Affiche
    Call DisBonjour(1)
End Sub
```

```
' Code de TstPublicPrivateModule2:
' ******************************

Option Explicit
Private Valeur As String

Private Sub Affiche()
    Cells(posY, 1) = "Attention, je suis dans la macro'' _
        & " ""Affiche"" du module 2."
    Cells(posY, 2) = "Je suis en ligne " & posY _
        & " grâce à ""posY"" qui est public."
    Cells(posY, 3) = "La ""Valeur"" private est: " & Valeur
    posY = posY + 1
End Sub

Sub ExecModule2(Optional dummy As Boolean)
' dummy ne sert qu'a empêcher que je n'apparaisse
' dans le liste des macros du menu outils, macros

    Valeur = "Un texte"
    Call Affiche
    Call DisBonjour(2)
End Sub
```

Remarques :

Famille : Programmation

Voir aussi : Public

Public

Type : Mots réservés

Syntaxe : Public définition_de_macro_ou_de_variable

Description : Rend disponible la macro ou la variable, dont la définition suit, n'importe où dans le programme.

Exemple :

Listing D.67 – Public

```
' Code de ThisWorkbook:
' **********************

Option Explicit

Sub process()

    posY = 1
    Call ExecModule1
    Call ExecModule2
End Sub
```

```
' Code de TstPublicPrivateModule1:
' ********************************

Option Explicit

Public posY As Long
Private Valeur As Single

Private Sub Affiche()
    Cells(posY, 1) = "Je suis dans la macro ""Affiche"" du'' _
        & "module 1."
    Cells(posY, 2) = "Je suis en ligne " & posY _
        & " grâce à ""posY"" qui est public."
    Cells(posY, 3) = "La ""Valeur"" private est: " & Valeur
    posY = posY + 1
End Sub

Public Sub DisBonjour(nbDeFois As Long)
    Dim n As Long
    For n = 1 To nbDeFois
        Cells(posY, 1) = "Bonjour depuis le module 1."
        posY = posY + 1
    Next n
End Sub

Sub ExecModule1(Optional dummy As Boolean)
' dummy ne sert qu'a empêcher que je n'apparaisse
' dans le liste des macros du menu outils, macros

    Valeur = 1.23456
    Call Affiche
    Call DisBonjour(1)
End Sub


' Code de TstPublicPrivateModule2:
' ********************************

Option Explicit
Private Valeur As String

Private Sub Affiche()
    Cells(posY, 1) = "Attention, je suis dans la macro'' _
        & " ""Affiche"" du module 2."
    Cells(posY, 2) = "Je suis en ligne " & posY _
        & " grâce à ""posY"" qui est public."
    Cells(posY, 3) = "La ""Valeur"" private est: " & Valeur
    posY = posY + 1
End Sub
```

```
Sub ExecModule2(Optional dummy As Boolean)
' dummy ne sert qu'a empêcher que je n'apparaisse
' dans le liste des macros du menu outils, macros

    Valeur = "Un texte"
    Call Affiche
    Call DisBonjour(2)
End Sub
```

Remarques :

Famille : Programmation

Voir aussi : Private

Randomize

Type : Mots Réservés

Syntaxe : Randomize [nombre]

Description : Initialise le générateur de nombres pseudo aléatoire Rnd. Si l'on ne fournit pas de « nombre », l'horloge de l'ordinateur est utilisée pour initialiser le générateur de nombres pseudo aléatoire. Pour obtenir à plusieurs reprises la même suite de nombres pseudo aléatoires, il faut faire un Rnd avec une valeur négative puis initialiser immédiatement Randomize avec une valeur.

Exemple :

- Randomize
- Randomize 2.3

Listing D.68 – Randomize

```
Sub TstRandomize()
    Dim PosY as long
    Dim N As Long

    PosY = 1
    Rnd (-1)
    Randomize 2.3
    For N = 1 To 10
        Cells(PosY, 1) = "Rnd()"
        Cells(PosY, 2) = Rnd()
        PosY = PosY + 1
    Next N

    Rnd (-1)
    Randomize 2.3
    For N = 1 To 10
        Cells(PosY, 1) = "Rnd()"
        Cells(PosY, 2) = Rnd()
        PosY = PosY + 1
    Next N
End Sub
```

Remarques :

Famille : Programmation

Voir aussi : Rnd

ReDim

Type : Mots réservés

Syntaxe : ReDim [Preserve] variable(liste_de_valeurs)

Description : Re-dimensionne ou réaffecte le tableau « variable ». Si le mot réservé `Preserve` est utilisé, les valeurs d'origines sont conservées, sinon elles sont remplacées ou supprimées.

Exemples :

Listing D.69 – ReDim

```
Sub TstReDim()
    Dim TableauTailleFixe(3)
    Dim TableauChaine() As String

    PosY = PosY + 1
    Cells(PosY, 1) = "Redim"
    PosY = PosY + 1

    TableauTailleFixe(0) = 0
    TableauTailleFixe(1) = 1
    TableauTailleFixe(2) = 2
    TableauTailleFixe(3) = 3

    ReDim TableauChaine(2)
    Cells(PosY, 1) = "ReDim TableauChaine(2)"
    PosY = PosY + 1
    TableauChaine(0) = "Texte1"
    TableauChaine(1) = "Texte2"
    TableauChaine(2) = "Texte3"
    Cells(PosY, 1) = "TableauChaine(2)"
    Cells(PosY, 2) = TableauChaine(2)
    PosY = PosY + 1

    ReDim TableauChaine(4)
    Cells(PosY, 1) = "ReDim TableauChaine(4)"
    PosY = PosY + 1
    TableauChaine(3) = "Texte4"
    TableauChaine(4) = "Texte5"
    Cells(PosY, 1) = "TableauChaine(2)"
    Cells(PosY, 2) = TableauChaine(2)
    PosY = PosY + 1
    Cells(PosY, 1) = "TableauChaine(3)"
    Cells(PosY, 2) = TableauChaine(3)
    PosY = PosY + 1
```

```
    ReDim Preserve TableauChaine(3)
    Cells(PosY, 1) = "ReDim Preserve TableauChaine(3)"
    PosY = PosY + 1
    TableauChaine(0) = "NouveauTexte1"
    TableauChaine(1) = "NouveauTexte2"
    TableauChaine(2) = "NouveauTexte3"
    Cells(PosY, 1) = "TableauChaine(2)"
    Cells(PosY, 2) = TableauChaine(2)
    PosY = PosY + 1
    Cells(PosY, 1) = "TableauChaine(3)"
    Cells(PosY, 2) = TableauChaine(3)
    PosY = PosY + 1
End Sub
```

Remarques : Lorsque l'on re-dimensionne un tableau à une dimension plus petite, les valeurs supérieures sont perdues.

Famille : Type de données

Voir aussi : Array

Rem

Type : Mots réservés

Syntaxe : Rem texte_de_commentaire

Description : Permet d'insérer des commentaires au sein du programme. Tout ce qui suit Rem jusqu'à la fin de la ligne est ignoré et considéré comme des commentaires. La forme abrégée de Rem est l'apostrophe.

Exemple :
- Rem Initialisation de toutes les variables
- ' Initialisation de toutes les variables

Remarques :

Famille : Programmation

Voir aussi :

Replace

Type : Fonction

Syntaxe : Replace(chaîne_originale, chaîne_de_recherche, chaîne_de_remplacement[, numéro_du_caractère_de_début_de_la_recherche [, nombre_de_remplacements]])

Description : Retourne la « chaîne_originale » dans laquelle la « chaîne_de_recherche » a été remplacée par la « chaîne_de_remplacement ». S'il est spécifié, on commence la recherche au caractère « numéro_du_caractère_de_début_de_la_recherche ». Si « nombre_de_remplacements » est indiqué, on ne fera que ce nombre de remplacement, même s'il existe d'autres occurrences dans la « chaîne_originale ».

Exemple :

Listing D.70 – Replace

```
Sub TstReplace()
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "Replace"
    PosY = PosY + 1
    Cells(PosY, 1) = "Replace(""anticonstitutionnel" _
        & "lement"", ""ti"", ""??"", 3, 2)"
    Cells(PosY, 2) = Replace("anticonstitutionnellement", _
        "ti", "??", 3, 2)
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Chaînes de caractères

Voir aussi :

Right

Type : Fonction

Syntaxe : Right(chaîne_de_caractères, nombre)

Description : Retourne le « nombre » de caractères indiqué qui correspondent à la partie droite de la « chaîne_de_caractères » fournie.

Exemple :

Listing D.71 – Right

```
Sub TstRight()
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "Right"
    PosY = PosY + 1
    Cells(PosY, 1) = "Right(""Programmer en VBA"", 5)"
    Cells(PosY, 2) = Right("Programmer en VBA", 5)
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Chaînes de caractères

Voir aussi : Left, Len, Mid

Rnd

Type : Fonction

Syntaxe : Rnd([réel_court])

Description : Retourne un nombre pseudo aléatoire compris entre 0 et 1. Si « réel_court » est négatif, on obtient toujours la même valeur, fonction de la valeur de « réel_court ». Si « réel_court » est égal à zéro, on obtient le dernier nombre pseudo aléatoire généré. Si « réel_court » est supérieur à zéro ou omis, on obtient le prochain nombre pseudo aléatoire de la série.

Exemple :

Listing D.72 – Rnd

```
Sub TstRnd()
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "Rnd"
    PosY = PosY + 1
    Cells(PosY, 1) = "Rnd()"
    Cells(PosY, 2) = Rnd()
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "Rnd()"
    Cells(PosY, 2) = Rnd()
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "Rnd(10)"
    Cells(PosY, 2) = Rnd(10)
    PosY = PosY + 1
    Cells(PosY, 1) = "Rnd(10)"
    Cells(PosY, 2) = Rnd(10)
    PosY = PosY + 1
    Cells(PosY, 1) = "Rnd(-10)"
    Cells(PosY, 2) = Rnd(-10)
    PosY = PosY + 1
    Cells(PosY, 1) = "Rnd(-10)"
    Cells(PosY, 2) = Rnd(-10)
    PosY = PosY + 1
End Sub
```

Remarques : Voir Randomize pour obtenir une suite de nombre pseudo aléatoire qui soit toujours la même.

Famille : Programmation

Voir aussi : Randomize

Round

Type : Fonction

Syntaxe : Round(Réel, [nombre_de_décimales])

Description : Retourne la valeur arrondie à la plus proche du « réel » fourni. « nombre_de_décimales » est facultatif et indique le nombre de décimales du résultat. S'il est omis, `Round` retourne un entier.

Exemple :

Listing D.73 – Round

```
Sub TstRound()
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "Round"
    PosY = PosY + 1
    Cells(PosY, 1) = "Round(2.36)"
    Cells(PosY, 2) = Round(2.36)
    PosY = PosY + 1
    Cells(PosY, 1) = "Round(2.96)"
    Cells(PosY, 2) = Round(2.96)
    PosY = PosY + 1
    Cells(PosY, 1) = "Round(-2.36)"
    Cells(PosY, 2) = Round(-2.36)
    PosY = PosY + 1
    Cells(PosY, 1) = "Round(-2.96)"
    Cells(PosY, 2) = Round(-2.96)
    PosY = PosY + 1
    Cells(PosY, 1) = "Round(1.12345,3)"
    Cells(PosY, 2) = Round(1.12345, 3)
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Mathématique

Voir aussi : Abs, Fix, Int

RTrim

Type : Fonction

Syntaxe : RTrim(chaîne_de_caractères)

Description : Retourne la « chaîne_de_caractères » après en avoir supprimé les espaces se trouvant à droite.

Exemple :

Listing D.74 – RTrim

```
Sub TstRTrim()
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "RTrim"
    PosY = PosY + 1
    Cells(PosY, 1) = "->RTrim(""    Programmer en VBA    "")<-"
    Cells(PosY, 2) = "->" & RTrim("    Programmer en VBA    ") _
    & "<-"
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Chaînes de caractères

Voir aussi : LTrim, Trim

Second

Type : Fonction

Syntaxe : Second(chaîne_de_caractères)

Description : Retourne les secondes extraite de la « chaîne_de_caractères » fournie qui représente des heures, minutes, secondes.

Exemple :

Listing D.75 – Second

```
Sub TstSecond()
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "Second"
    PosY = PosY + 1
    Cells(PosY, 1) = "Second(""10:23:52"")"
    Cells(PosY, 2) = Second("10:23:52")
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Date/Heure

Voir aussi : Hour, Minute

SelectCase

Type : Mot réservé

Syntaxe : Select Case valeur_à_tester Case valeur(s)_de_référence instructions Case valeur(s)_de_référence instructions ... Case Else instructions End Select

Description : Effectue des tests sur « valeur_à_tester ». Le bloc d'instructions défini pour chaque « valeur_de_référence » est exécuté lorsque la valeur correspond, sinon le bloc d'instructions correspondant au `Case Else` est exécuté, s'il existe.

Exemple :

```
Sub TstSelectCase
    Dim I As Integer

    For I = 1 To 10
        Cells(I + 13, 1) = I

        Select Case I
            Case 1
                Cells(I + 13, 2) = "I à la valeur 1"
            Case 2
                Cells(I + 13, 2) = "I à la valeur 2"
            Case 8 To 10
                Cells(I + 13, 2) = "I est entre 8 et 10"
            Case Else
                Cells(I + 13, 2) = "I est supérieur à 2"
        End Select
    Next I
End Sub
```

Remarques :

Famille : Programmation

Voir aussi : If ... Then ... Else

Set

Type : Mots réservés

Syntaxe : set variable_objet = objet

Description : Assigne un objet à « variable_objet ». C'est indispensable lorsque l'on va vouloir manipuler Powerpoint depuis Excel, par exemple.

Exemple :

Listing D.76 – Set

```
Sub TstSet()
    ' Ne pas oublier dans l'éditeur VBA d'aller cocher:
    ' Outils -> Références ->
    ' Microsoft Powerpoint 10.0 Object library
```

```
    Dim ppApplication As PowerPoint.Application
    Dim ppSlide As PowerPoint.Slide

    ' Récupère Powerpoint s'il est ouvert
    Set ppApplication = GetObject("PowerPoint.Application")
    ' Ouvre Powerpoint s'il n'est pas ouvert
    If ppApplication is Nothing then
        Set ppApplication = New PowerPoint.Application
    End If
    ' Le rend visible
    ppApplication.Visible = True
    ' Ajoute une nouvelle présentation
    ppApplication.Presentations.Add
    ' Ajoute une nouvelle diapositive de titre
    ppApplication.ActivePresentation.Slides.Add _
        1, ppLayoutTitle
    ' Ajoute une nouvelle diapositive
    ' titre + contenu texte à la fin
    ppApplication.ActivePresentation.Slides.Add _
        ppApplication.ActivePresentation.Slides.Count + 1,
        ppLayoutText
    ' Revient à la première diapositive
    ppApplication.ActiveWindow.View.GotoSlide 1
    ' Active Powerpoint
    AppActivate ("Microsoft PowerPoint")
    ' Fait le ménage dans les variables
    Set ppApplication = Nothing
End Sub
```

Remarques :

Famille : Programmation

Voir aussi :

Sgn

Type : Fonction

Syntaxe : Sgn(nombre)

Description : Retourne le signe du « nombre » fourni. 1 s'il est positif, -1 dans le cas contraire.

Exemple :

Listing D.77 – Sgn

```
Sub TstSgn()
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "Sgn"
    PosY = PosY + 1
```

```
    Cells(PosY, 1) = "Sgn(2.36)"
    Cells(PosY, 2) = Sgn(2.36)
    PosY = PosY + 1
    Cells(PosY, 1) = "Sgn(-2.36)"
    Cells(PosY, 2) = Sgn(-2.36)
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Mathématique

Voir aussi :

Sin

Type : Fonction

Syntaxe : Sin(angle_en_radians)

Description : Retourne le sinus de l'« angle_en_radians » fournit. L'angle est en radians.

Exemple : Sin(0.523598) retourne 0.499999

Listing D.78 – Sin

```
Sub TstSin()
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "Sin"
    PosY = PosY + 1
    Cells(PosY, 1) = "Sin(0.523598)"
    Cells(PosY, 2) = Sin(0.523598)
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Mathématique

Voir aussi : Cos, Tan, Atn

Single

Type : Mot réservé

Syntaxe : variable As Single

Description : Défini la « variable » comme étant de type `single`, c'est à dire acceptant les valeurs allant de -3,402823 E38 à 3,402823 E38.

Exemple : Dim resultat As Single

Remarques :

Famille : Programmation

Voir aussi : CSng,

Space

Type : Fonction

Syntaxe : Space(nombre_d'espaces)

Description : Construit une chaîne de caractères contenant le « nombre_d'espaces » spécifié.

Exemple :

Listing D.79 – Space

```
Sub TstSpace()
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "Space"
    PosY = PosY + 1
    Cells(PosY, 1) = "->Space(5)<-"
    Cells(PosY, 2) = "->" & Space(5) & "<-"
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Chaînes de caractères

Voir aussi : concaténation, String

Split

Type : Fonction

Syntaxe : Split(chaîne_de_caractères, séparateur)

Description : Construit un tableau qui contient chacun des morceaux de la « chaîne_de_caractères », précédemment séparés les uns des autres par le « séparateur ».

Exemple :

Listing D.80 – Split

```
Sub TstSplit()

    Dim resultat As Variant

    PosY = PosY + 1
    Cells(PosY, 1) = "Split"
    resultat = Split("Aimer, lire, apprendre", ",")
    PosY = PosY + 1
    Cells(PosY, 1) = "resultat(0)" & Chr(10) & _
      "avec resultat = Split(""Aimer, lire, apprendre"",""," _
      & " "","")"
    Cells(PosY, 2) = resultat(0)
    PosY = PosY + 1
    Cells(PosY, 1) = "resultat(1)" & Chr(10) & _
```

```
      "avec resultat = Split(""Aimer, lire, apprendre"","" _
      & "" "","")"
    Cells(PosY, 2) = resultat(1)
    PosY = PosY + 1
    Cells(PosY, 1) = "resultat(2)" & Chr(10) & _
      "avec resultat = Split(""Aimer, lire, apprendre"","" _
      & "" "","")"
    Cells(PosY, 2) = resultat(2)
    PosY = PosY + 1
End Sub
```

Remarques : Attention, le tableau obtenu commence à l'indice 0.

Famille : Chaînes de caractères

Voir aussi : concaténation

Sqr

Type : Fonction

Syntaxe : Sqr(entier)

Description : Retourne la racine carré de l'« entier » fourni.

Exemple :

Listing D.81 – Sqr

```
Sub TstSqr()
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "Sqr"
    PosY = PosY + 1
    Cells(PosY, 1) = "Sqr(3)"
    Cells(PosY, 2) = Sqr(3)
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Mathématique

Voir aussi : puissance

StrComp

Type : Fonction

Syntaxe : StrComp(chaîne1, chaîne2)

Description : Compare les chaînes de caractères « chaîne1 » et « chaîne2 ». Retourne 0 si elles sont identiques, -1 dans le cas contraire.

Exemple :

Listing D.82 – StrComp

```
Sub TstStrComp()
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "StrComp"
    PosY = PosY + 1
    Cells(PosY, 1) = "StrComp(""Programmer en VBA""," _
      & " ""Programmer en VBA"")"
    Cells(PosY, 2) = StrComp("Programmer en VBA", _
      "Programmer en VBA")
    PosY = PosY + 1
    Cells(PosY, 1) = "StrComp(""Programmer en VBA""," _
      & " ""Programmez en VBA"")"
    Cells(PosY, 2) = StrComp("Programmer en VBA", _
      "Programmez en VBA")
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Chaînes de caractères

Voir aussi : InStr, InStrRev

String

Type : Fonction

Syntaxe : String(nombre, chaîne_de_caractères)

Description : Retourne une chaîne de caractères constitué de « nombre » de fois la « chaîne_de_caractères » fournie.

Exemple :

Listing D.83 – String

```
Sub TstString()
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "String"
    PosY = PosY + 1
    Cells(PosY, 1) = "String(13, ""a"")"
    Cells(PosY, 2) = String(13, "a")
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Chaînes de caractères

Voir aussi : concaténation, Space

StrReverse

Type : Fonction

Syntaxe : StrReverse(chaîne_de_caractères)

Description : Retourne une chaîne de caractères composée des caractères de la « chaîne_de_caractères » originale mais dont l'ordre a été inversé (le dernier caractère devient le premier et vice-versa).

Exemple :

Listing D.84 – StrReverse

```
Sub TstStrReverse()
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "StrReverse"
    PosY = PosY + 1
    Cells(PosY, 1) = "StrReverse(""Programmer en VBA"")"
    Cells(PosY, 2) = StrReverse("Programmer en VBA")
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Chaînes de caractères

Voir aussi :

Sub

Type : Mots réservés

Syntaxe : Sub Nom_de_macro ... End Sub

Description : Déclare une nouvelle macro en lui donnant un nom. Le contenu de la macro se trouve entre cette déclaration et le mot clé `End Sub`.

Exemple :

Listing D.85 – Sub

```
Dim PosY As Long

Sub AutoAdjust(aSheetName As String)
' adjust all cells width and height
    Sheets(aSheetName).Select
    Cells.Select
    Cells.EntireColumn.AutoFit
    Cells.EntireRow.AutoFit
    Range("A1").Select
End Sub

Sub TstSub()
    ' Nomme la feuille
```

```
    ActiveSheet.Name = "Test"

    PosY = PosY + 1
    Cells(PosY, 1) = "Sub"
    PosY = PosY + 1
    Cells(PosY, 1) = "Call AutoAdjust(""Test"")"
    Call AutoAdjust("Test")"
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Programmation

Voir aussi : Call, Function

Tan

Type : Fonction

Syntaxe : Tan(angle_en_radians)

Description : Retourne la tangente de l'« angle_en_radians » fournit. L'angle est en radians.

Exemple : Tan(0.523336) retourne 0.576999

Listing D.86 – Tan

```
Sub TstTan()
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "Tan"
    PosY = PosY + 1
    Cells(PosY, 1) = "Tan(0.523336)"
    Cells(PosY, 2) = Tan(0.523336)
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Mathématique

Voir aussi : Atn, Cos, Sin

Time

Type : Fonction

Syntaxe : Time()

Description : Retourne l'heure courante.

Exemple :

Listing D.87 – Time

```
Sub TstTime()
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "Time"
    PosY = PosY + 1
    Cells(PosY, 1) = "Time()"
    Cells(PosY, 2) = Time()
    Dim PosY As Long

    PosY = 1
End Sub
```

Remarques :

Famille : Date/Heure

Voir aussi : Date, Now

TimeSerial

Type : Fonction

Syntaxe : TimeSerial(heures, minutes, secondes)

Description : Retourne l'heure complète correspondant aux « heures », « minutes » et « secondes » fournies.

Exemple :

Listing D.88 – TimeSerial

```
Sub TstTimeSerial()
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "TimeSerial"
    PosY = PosY + 1
    Cells(PosY, 1) = "TimeSerial(10,23,52)"
    Cells(PosY, 2) = TimeSerial(10, 23, 52)
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Date/Heure

Voir aussi : TimeValue

TimeValue

Type : Fonction

Syntaxe : TimeValue(chaîne_de_caractères_hh :mm :ss)

Description : Retourne l'heure complète correspondant à la « chaîne_de_caractères_hh :mm :ss » fournie.

Exemple :

Listing D.89 – TimeValue

```
Sub TstTimeValue()
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "TimeValue"
    PosY = PosY + 1
    Cells(PosY, 1) = "TimeValue(""10:23:52"")"
    Cells(PosY, 2) = TimeValue("10:23:52")
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Date/Heure

Voir aussi : TimeSerial

Trim

Type : Fonction

Syntaxe : Trim(chaîne_de_caractères)

Description : Retourne la « chaîne_de_caractères » après en avoir supprimé les espaces se trouvant à gauche et à droite.

Exemple :

Listing D.90 – Trim

```
Sub TstTrim()
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "Trim"
    PosY = PosY + 1
    Cells(PosY, 1) = "->Trim(""   Programmer en VBA   "")<-"
    Cells(PosY, 2) = "->" & Trim("   Programmer en VBA   ") _
      & "<-"
    PosY = PosY + 1
End Sub
```

Remarques :

Famille :

Voir aussi : LTrim, RTrim

True

Type : Mot réservé

Syntaxe : True

Description : Type booléen valant vrai (-1). On l'utilise beaucoup dans tout ce qui est test, comparaison.

Exemple :

```
Sub TstTrue()
    Dim Trouve As Boolean

    Trouve = True
    PosY = PosY + 1
    Cells(PosY, 1) = "True"
    PosY = PosY + 1
    Cells(PosY, 1) = "if Trouve = True then" _
      & " Cells(PosY,2) = ""Trouvé"""
    If Trouve = True Then
        Cells(PosY, 2) = "Trouvé"
    Else
        Cells(PosY, 2) = "Perdu"
    End If
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Type de données

Voir aussi : False

UBound

Type : Fonction

Syntaxe : UBound(tableau)

Description : Retourne le plus grand indice du « tableau ».

Exemple :

Listing D.91 – UBound

```
Sub TstUBound()
    Dim liste As Variant
    Dim PosY As Long

    liste = Array("Aimer", "apprendre", "lire")
    PosY = 1
    Cells(PosY, 1) = "UBound"
    PosY = PosY + 1
    Cells(PosY, 1) = "UBound(liste) -> " _
      & "liste(UBound(liste))" & Chr(10) _
```

```
      & "avec liste = Array(""Aimer"", " _
      & """apprendre"", ""lire"")"
    Cells(PosY, 2) = UBound(liste) & " -> " _
      & liste(UBound(liste))
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Tableau

Voir aussi : LBound

UCase

Type : Fonction

Syntaxe : UCase(chaîne_de_caractères)

Description : Retourne la « chaîne_de_caractères » fournie, convertie en majuscules.

```
Sub TstUCase()
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "UCase"
    PosY = PosY + 1
    Cells(PosY, 1) = "UCase(""TexTe A MetTrE EN MAjusCule"")"
    Cells(PosY, 2) = UCase("TexTe A MetTrE EN MAjusCule")
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Chaînes de caractères

Voir aussi : LCase

Variant

Type : Mot réservé

Syntaxe : variable As Variant

Description : Défini la « variable » comme étant de type `variant`, c'est à dire s'adaptant à sont contenu.

Exemple : Dim resultat As Variant

Remarques :

Famille : Programmation

Voir aussi :

VarType

Type : Fonction
Syntaxe : VarType(variable)
Description : Retourne, sous la forme d'un entier, le type de la « variable » fournie.

Constante prédéfinie	Valeur	Description
vbEmpty	0	Variable non initialisée
vbNull	1	Variable ne contenant pas de donnée valide
vbInteger	2	Entier
vbLong	3	Entier long
vbSingle	4	Réel simple
vbDouble	5	Réel double
vbCurrency	6	Monétaire
vbDate	7	Date
vbString	8	Chaîne de caractères
vbObject	9	Objet
vbError	10	Variable d'erreur
vbBoolean	11	Booléen
vbVariant	12	Variant
vbByte	17	Octet
vbUserDefinedType	36	Défini par l'utilisateur
vbArray	8192	Tableau

TAB. D.5 – Valeurs pour les types de variables

Exemple :

```
Sub TstVarType()
    Dim valeur As Variant
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "VarType"
    PosY = PosY + 1
    valeur = 123
    Cells(PosY, 1) = "VarType(valeur) avec valeur = 123"
    Cells(PosY, 2) = VarType(valeur)
    PosY = PosY + 1
    valeur = 123.123
    Cells(PosY, 1) = "VarType(valeur) avec valeur = 123.123"
    Cells(PosY, 2) = VarType(valeur)
    PosY = PosY + 1
    valeur = "123.123"
    Cells(PosY, 1) = "VarType(valeur) avec" _
      & " valeur = ""123.123"""
    Cells(PosY, 2) = VarType(valeur)
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Programmation

Voir aussi : Array, Boolean, Byte, Currency, Date, Double, Integer, Long, Single, String, Variant

WeekDay

Type : Fonction

Syntaxe : WeekDay(chaîne_de_caractères_date [, premier_jour_de_la_semaine])

Description : Retourne le numéro du jour dans la semaine. « premier_jour_de_la_semaine », s'il est fourni, défini le jour qui est considéré comme le premier de la semaine.

Constante prédéfinie	Valeur	Description
vbUseSystem	0	Utilise la configuration de l'ordinateur
vbSunday	1	Dimanche (valeur par défaut)
vbMonday	2	Lundi
vbTuesday	3	Mardi
vbWednesday	4	Mercredi
vbThursday	5	Jeudi
vbFriday	6	Vendredi
vbSaturday	7	Samedi

TAB. D.6 – Valeurs pour les jours de la semaine

Exemple :

```
Sub TstWeekday()
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "Weekday"
    PosY = PosY + 1
    Cells(PosY, 1) = "Weekday(""25/12/2005"")"
    Cells(PosY, 2) = Weekday("25/12/2005")
    PosY = PosY + 1
End Sub
```

Remarques :

Famille :

Voir aussi : WeekDayName

WeekDayName

Type : Fonction

Syntaxe : WeekdayName(jour [, court])

Description : Retourne le nom du « jour » fourni. Si « court » vaut True, alors on obtient la forme abrégée du nom du jour.

Exemple :

Listing D.92 – WeekdayName

```
Sub TstWeekdayName()
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "WeekdayName"
    PosY = PosY + 1
    Cells(PosY, 1) = "WeekdayName(Weekday(""25/12/2005""))"
    Cells(PosY, 2) = WeekdayName(Weekday("25/12/2005"))
    PosY = PosY + 1
    Cells(PosY, 1) = "WeekdayName(Weekday(""25/12/2005""),"_
      & " True)"
    Cells(PosY, 2) = WeekdayName(Weekday("25/12/2005"), True)
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Date/Heure

Voir aussi : MonthName, WeekDay

While ... Wend

Type : Mot réservé

Syntaxe : While condition instructions Wend

Description : Exécute les « instructions » comprises entre While et Wend tant que la condition est remplie. La « condition » est testée en début de boucle.

Exemple :

Listing D.93 – While ... Wend

```
Sub TstWhileWend
    Dim I As Integer
    Dim J As Integer

    I = 2
    J = 1
    While I < 300
        cells(1,J)= I
        J = J + 1
        I = I * I
    Wend
End Sub
```

Remarques :

Famille : Programmation

Voir aussi : Do . . . Loop Until, Do . . . Loop While, For . . . Next, For Each . . . Next

With

Type : Mots réservés

Syntaxe : With objet instructions End With

Description : Regroupe une série d'« instructions » qui affectent sur un même objet. Toutes les actions commençant par un point entre `With` et `End With` seront appliquées à « objet ». On utilisera beaucoup cette possibilité lorsqu'il s'agira de changer l'apparence (format, police, positionnement . . .) de certains éléments d'Excel.

Exemple :

Listing D.94 – With

```
Sub TstWith()

    ' Efface le contenu des cellules
    Cells.Select
    Selection.Clear
    ' Formate les cellules
    With Selection
        .NumberFormat = "@"
        .Font.Bold = True
        .HorizontalAlignment = xlCenter
        .VerticalAlignment = xlCenter
    End With
    Range("A1").Select
End Sub
```

Remarques :

Famille : Programmation

Voir aussi :

Xor

Type : Opérateur

Syntaxe :

1. condition1 Xor condition2
2. entier1 Xor entier2

Description :

1. Ou logique exclusif, renvoi True (vrai) si une des deux conditions, et une seule, est remplie.
2. Ou binaire exclusif, dans l'entier renvoyé, les bits à 1 sont ceux qui était à 1 dans « entier1 » ou dans « entier2 », mais pas dans les deux. Les autres sont à 0.

Exemples :

1. If (A = 5) Xor (B = 22) Then "Vrai"
2. 5 Xor 22 retourne 19 (00000101 And 00010110 donne 00010011)

```
Sub TstXor()
    Dim A As Long
    Dim B As Long
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "Xor"
    A = 5
    B = 22
    PosY = PosY + 1
    Cells(PosY, 1) = "(A = 5) Xor (B = 22)"
    If (A = 5) Xor (B = 22) Then
        Cells(PosY, 2) = "Vrai"
    Else
        Cells(PosY, 2) = "Faux"
    End If
    PosY = PosY + 1
    Cells(PosY, 1) = "5 Xor 22"
    Cells(PosY, 2) = 5 Xor 22
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Opérateur logique

Voir aussi : And, Not, Xor

Year

Type : Fonction

Syntaxe : Year(chaîne_de_caractères)

Description : Retourne l'année extraite de la date fournie sous forme d'une « chaîne_de_caractères ».

Exemple :

```
Sub TstYear()
    Dim PosY As Long

    PosY = 1
    Cells(PosY, 1) = "Year"
    PosY = PosY + 1
    Cells(PosY, 1) = "Year(""25/12/2005"")"
    Cells(PosY, 2) = Year("25/12/2005")
    PosY = PosY + 1
End Sub
```

Remarques :

Famille : Date/Heure

Voir aussi : DatePart, Day, Month

Glossaire

API	interface de programmation de l'application (Application Programming Interface)
Code	texte qui contient toutes les instructions d'un programme
Excel	tableur faisant partie de la suite Microsoft Office
Module	sous-ensemble de macros et fonctions que l'on a rassemblé
VBA	Visual Basic for Application, langage de programmation basic pour les applications

Table des figures

Liste des tableaux

Listings

Index

Référence du langage par ordre alphabétique

Référence du langage par type

Référence du langage par famille

Sur le web

[1] Daniel Josserand, VBAXL, http ://dj.joss.free.fr/, 1996

[2] Daniel Josserand, FAQ du forum Microsoft.Public.Fr.Excel, http ://dj.joss.free.fr/faq.htm, 2000

[3] Misange, Excelabo, http ://www.excelabo.net/, 2000

[4] Catherine Monier, Cathy astuce, http ://www.cathyastuce.com/vba.htm, 2000

[5] Des passionnés, Excel downloads, http ://www.excel-downloads.com/, 2000

[6] Laurent Longre, X-Cell, http ://xcell05.free.fr/, 1999

[7] Joseph Rubin, Exceltip, http ://www.exceltip.com/,

[8] Jon Peltier, Jon Peltier's Excel Page, http ://peltiertech.com/Excel/index.html,

[9] Tech on the Net, Excel Topics, http ://www.techonthenet.com/excel/index.php,