9.2.2 CURRENCY

Les variables de type **Currency** donne un nombre à virgule fixe comprenant 15 chiffres à gauche du séparateur décimal et 4 chiffres à droite.
Cette représentation offre une plage comprise entre
`-922 337 203 685 477,5808  et  922 337 203 685 477,5807`
Le type de donnée Currency est extrêmement utile pour les calculs monétaires et les calculs à virgule fixe pour lesquels la précision revêt une importance particulière.

9.2.3 DATE ET HEURE

Les variables de type **Date** représentent des dates comprises **entre le 1er janvier 100 et le 31 décembre 9999** et des **heures allant de 0:00:00 à 23:59:59**. Toute valeur reconnue comme date littérale peut être affectée à une variable de type Date.
Les dates littérales doivent être délimitées par des signes dièse (#). Par exemple, `#January 1, 1993#` ou `#1 jan 93#`. Une fois reconnue par le système (**mais pensez que VBA ne reconnaît que l'anglais**), la date est interprétée et réécrite dans VBE sous la forme #Mois/Jour/Année#.
De la même façon, les heures sont reconnues et transcrite dans VBE dans le sytème US.
Exemples :

Ce que vous saisissez dans VBE	Ce qui restera écrit, après validation, dans VBE
`varDH = #13 jan 2000# '` les 3 premières lettres suffisent pour le mois	`varDH = #1/13/2000#`
`varDH = #17 feb# '` L'année courante est prise par défaut	`varDH = #2/17/2005#`
`varDH = #5 APR 15:29:22#` ' forme date heure minute seconde	`varDH = #4/5/2005 3:29:22 PM#`
`varDH = #3:0#`	`varDH = #3:00:00 AM#`
`varDH = #January 12, 1993#`	`varDH = #1/12/1993#`
`varDH = #5 7 9#`	`varDH = #5/7/2009#`
`varDH = #23,8,47# '` **attention !**	`varDH = #8/23/1947#`
`varDH = #2,8,47# '` **attention !**	`varDH = #2/8/1947#`

Les variables de type Date affichent les dates au format de date abrégé reconnu par votre ordinateur. Les heures s'affichent au format horaire (sur une plage de 12 ou de 24 heures) reconnu par votre ordinateur.
Lorsque d'autres types de données numériques sont convertis en données de type Date, les valeurs situées à gauche du séparateur décimal représentent la date, tandis que les valeurs situées à droite correspondent à l'heure. Minuit est représenté par 0 et midi par 0,5. Les nombres entiers négatifs représentent des dates antérieures au 30 décembre 1899. Lorsque des variables de type Date sont converties dans un autre type de donnée numérique, elles se présentent uniquement sous la forme de nombres.

9.2.4 DOUBLE

Les variables de type **Double** (à virgule flottante en double précision).
La valeur peut être comprise entre
-1,79769313486232 E308 et -4,94065645841247 E-324 pour les valeurs négatives, et entre
4,94065645841247 E-324 et 1,79769313486232 E308 pour les valeurs positives.

9.2.5 INTEGER*

Entier dont la valeur peut être comprise entre -32 768 et 32 767
(Attention : ces valeurs qui sont déjà importantes suffisent à compter les 256 colonnes d'une feuille Excel, mais pas les 65 535 lignes…)

9.2.6 LONG*

Entier dont la valeur peut être comprise entre -2 147 483 648 et 2 147 483 647

9.2.7 LONGLONG*

Entier stockée en tant que nombres signés 64 bits (8 octets) dans une plage de valeurs allant de -9 223 372 036 854 775 808 à 9 223 372 036 854 775 807. Le caractère de déclaration de type pour les variables **LongLong** est l'accent circonflexe (^). **LongLong** est un type déclaré valide uniquement sur les plateformes 64 bits.

9.2.8 LONGPTR

Entier de type Long sur des systèmes 32 bits, entier de type LongLong sur des systèmes 64 bits) sont stockées en tant que nombres signés 32 bits (4 octets) dans une plage de valeurs allant de -2 147 483 648 à 2 147 483 647 sur des systèmes 32 bits, et en tant que nombres signés 64 bits (8 octets) dans une plage de valeurs allant de -9 223 372 036 854 775 808 à 9 223 372 036 854 775 807 sur des systèmes 64 bits.

9.2.9 SINGLE

Les variables de type **Single** (à virgule flottante en simple précision)
La valeur peut être comprise entre -3,402823 E38 et -1,401298 E-45 pour les valeurs négatives, et entre 1,401298 E-45 et 3,402823 E38 pour les valeurs positives.

9.2.10 STRING

(String veut dire chaîne en anglais). Il existe deux types de chaînes :

- Les chaînes de longueur variable, qui peuvent contenir jusqu'à 2 millions (2^{31}) de caractères.
- Les chaînes de longueur fixe qui comprennent un nombre de caractères déclarés d'environ 64 000 (2^{16}) caractères. Dans ce cas la syntaxe indiquant le nombre caractères sera de la forme :
  `Dim•varNom•As•String•*•5•'` VarNom aura toujours 5 caractères
  `varNom•=•"Marsupilami"•'` VarNom contient alors Marsu
  `varNom•=•"Moi"•'` VarNom contient alors Moi•• (avec 2 espaces à la fin)
  (voir Exemple 3 en fin de ce chapitre)

9.2.11 VARIANT

Le type de donnée Variant est le type de donnée par défaut, attribué à toutes les variables qui ne sont pas explicitement déclarées.

Le type de donnée **Variant** est un type de données particulier qui peut contenir des données de toutes sortes ainsi que les valeurs spéciales **Empty** (vide), **Error** (erreur) et **Null**

* Une variable de type **Integer** ou **Long** n'acceptera pas d'être déclarée directement avec une valeur non entière. Par contre si elle reçoit cette valeur pendant l'exécution du code, elle l'arrondira.

(expression rarement traduite car nul en français est trop facilement assimilé à **zéro** ou au mieux à **rien** ou **vide** alors que **Null signifie une non correspondance, une inadaptation au contexte**).
Vous pouvez utiliser le type de donnée Variant à la place d'un type de donnée fondamental lorsque vous recherchez plus de souplesse dans le traitement des données. Si vous ne spécifiez pas de type de données lorsque vous déclarez une variable, ou si vous ne la déclarez pas avant son utilisation, Visual Basic attribue automatiquement le type de données Variant.

9.2.12 POURQUOI NE PAS TOUJOURS UTILISER LE TYPE VARIANT ?

Au risque de faire hurler les puristes, pourquoi pas !
Mais il faut savoir que chaque type attribué occupe un certain espace mémoire. Si vous déclarez par exemple une variable avec le type Integer, vous n'occuperez qu'un petit espace de 2 octets mais si vous le faites avec le type variant l'espace occupé sera de 16 octets + 1 octet pour chaque caractère.
Le nombre 12 avec le type Integer nécessite 2 octets alors que le même nombre 12 avec le type variant nécessitera 16 + 2 = 18 octets.
Il ne suffit donc pas de déclarer les variables mais il faut de préférence mettre le bon type pour occuper le moins d'espace mémoire possible et ainsi faciliter le traitement. Cela sera d'autant plus nécessaire que l'application utilisera de nombreuses variables ou des variables matricielles conséquentes.

9.2.13 CARACTÈRE DE DÉCLARATION DE TYPE

Ce caractère, ajouté à un nom de variable, indique le type de données de la variable. Ce caractère n'existe pas pour tous les types de variable.

Integer	**%**	Currency	@	Double	**#**
Long	**&**	Single	**!**	String	**$**

La syntaxe de déclaration devient alors :

```
Dim varVal% ' varVal est de type Integer
Dim varMot$ ' varMOT est de type String
varVal = 22 ' le caractère de déclaration de type ne doit
varMot = "abcde" ' plus être utilisé après
```

Attention :

```
Dim varNom As String * 5 ' fonctionne
Dim varNom$ * 5 ' à ce jour ne fonctionne pas, j'ai essayé !
```

9.3 VARIABLE MATRICIELLE OU TABLEAU

Dans un premier temps, on a assimilé une variable à une boîte. Pour ce qui est des variables matricielles (appelées aussi variables tableaux, voire simplement tableaux), l'analogie pourrait être une boîte avec des compartiments tous identiques et du même type qu'un de ceux précédemment définis.
Pour continuer avec cette image, on peut également imaginer une boîte à plusieurs niveaux, chaque niveau ayant plus ou moins de compartiments. On aura alors un tableau à plusieurs **dimensions**, dimensions étant pris au sens mathématique du terme, c'est-à-dire que nous pourrons aller au-delà des trois dimensions perçues dans l'espace de notre environnement.

9.3.1 INDEXATION DES VARIABLES MATRICIELLES

Attention : par défaut le départ de l'indexation des variables se fait à partir de 0 ce qui n'est pas le cas des autres collections d'objets.
Pour démarrer à 1, il faut déclarer Option Base 1 en dehors des macros, en début de Module ou déclarer la limite inférieure et supérieure de l'indexation (voir dans l'exemple suivant).

9.3.2 SYNTAXE D'UNE VARIABLE MATRICIELLE

L'argument facultatif **subscripts** de l'instruction **Dim** donne les dimensions (jusqu'à 60) d'une variable matricielle (voir aide en ligne de l'instruction Dim pour, si nécessaire, connaître la syntaxe complète de la déclaration d'une variable) et les limites de son indexation.
Exemples de déclaration à une dimension :

Code dans deux module d'un même projet

Code du Module1

```
Option base 0 ' Par défaut quand non exprimé.
Sub Essai_Module1()
'
Dim varTableau1(3) As Date ' Tableau
' à 1 dimension et 4 éléments (0, 1, 2 ,3) de type date
'
Dim varTableau2(2 To 4) As Date ' Tableau
' à 1 dimension et 3 éléments (2, 3, 4) de type date
- - - - -
End Sub
```

limite inférieure de l'indexation
limite supérieure de l'indexation

Code du Module2

```
Option base 1
Sub Essai_Module2()
'
Dim varTableau1(3) As Date ' Tableau
' à 1 dimension et 3 éléments (1, 2, 3) de type date
'
Dim varTableau2(2 To 4) As Date ' Tableau
' à 1 dimension et 3 éléments (2, 3, 4) de type date
- - - - -
End Sub
```

Les exemples comparés ci avant montrent qu'il est référable d'exprimer les deux limites inférieure et supérieure de l'indexation, ceci **levant toute ambiguïté quelque soit l'Option Base retenue**.
Il n'est pas obligatoire de déclarer à sa création la dimension d'un tableau, mais il faudra le faire au moment de l'utiliser à l'aide de l'instruction ReDim (voir aide en ligne).
Exemple de déclaration à plusieurs dimensions :

```
Sub Essai_4Dimensions()
Dim varT(2 To 5, 1 To 7, 3 To 6, 1 To 2) As Integer
           4       7       4       2   ' nombre d'éléments
' VarT pourra prendre 4×7×4×2 = 224 valeurs
- - - - - -
End Sub
```

Pour attribuer une valeur à un des éléments d'une variable tableau, il faut indiquer l'index de chaque dimension. Exemple :

```
Sub Essai_Attribution()
Dim varRep As Integer, varT1() As Integer
Dim varT2(1 To 3) As Integer
Dim varT3(1 To 4, 1 To 3, 1 To 2) As Date
varT2(2) = 25 ' Charge 25 dans le 2e élément de varT2
varT3(4, 2, 2) = #7/14/2000# ' charge l'élément 4-2-2 de varT3
' à la date du 14 juillet 2000 (forme US Mois/Jour/Année dans VBA)
' Illustration des valeurs de varT1
ReDim varT1(3)
  For i = 1 To 3
    varT1(i) = Int((100 * Rnd) + 1) ' génération aléatoire
     MsgBox "varT1(" & i & ")=" & varT1(i)
  Next i
' Illustration des valeurs de varT2
  For i = 1 To 3
    MsgBox "varT2(" & i & ")=" & varT2(i)
  Next i
' Illustration des valeurs de varT3
  For i = 4 To 1 Step -1
    For j = 3 To 1 Step -1
      For k = 2 To 1 Step -1
         varRep = MsgBox("varT3( " & i & ", " _
         & j & ", " & k & ")=" & Format(varT3(i, j, k), _
         "d mmmm yyyy"), vbOKCancel, _
         " REMARQUEZ  BIEN  LA  DATE")
         If varRep = vbCancel Then Exit Sub ' abrège ...
      Next k
    Next j
  Next i
End Sub
```

9.4 VARIABLE DE TYPE OBJET

Une variable peut contenir un objet. Par analogie avec une variable classique qui a pu être comparée à une boîte plus ou moins grande, une variable-objet* est toujours une boîte mais dont la forme, la couleur, la contenance, etc… sont identiques à l'objet qu'elle représente. Cette variable-objet récupérera donc les propriétés et méthodes du type d'objet défini.

9.4.1 SYNTAXES DE LA DÉCLARATION ET DE L'UTILISATION D'UNE VARIABLE-OBJET

Pour la déclaration, le nom du type de l'objet vient en lieu et place du type de la valeur remplacée.

Exemple : `Dim varFeuilleCalcul As Worksheet`

Pour affecter une référence d'objet à une variable, vous devrez utiliser le mot **`Set` devant le nom de la variable-objet** placé avant le signe égal.

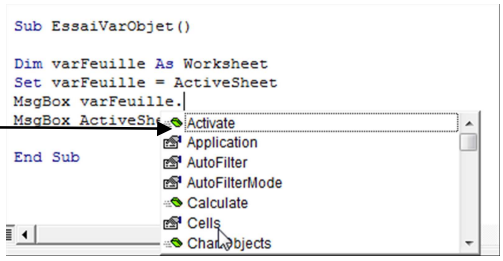
`Set varFeuilleCalcul = ActiveSheet`

Cet artifice d'écriture peut nous étonner (il est très fréquent dans les exemples de l'aide en ligne) alors même que l'élément remplacé pourrait être exprimé directement.

* **varaible-objet** est bien une entité, et c'est pourquoi il a été mis un trait d'union.

Pourquoi par exemple définir une variable qui va représenter la feuille active alors même que l'expression ActiveSheet suffit à désigner l'objet visé de façon unique ?!
C'est qu'en "cataloguant" ainsi une variable avec un type d'objet, celle-ci est reconnue comme objet de cette catégorie et se voit proposer dynamiquement les options correspondantes (dès que l'on tape le point, la liste de choix possible apparaît).
Pour illustrer, vous pouvez faire l'expérience suivante. Dans **VBE**, ouvrez une nouvelle macro et valider son nom (par exemple EssaiVarObjet) pour avoir End Sub de généré. Saisissez alors les lignes suivantes en observant bien comment vous sont proposées les arguments possibles au fur et à mesure de votre saisie :

```
Sub EssaiVarObjet()
Dim varFeuille As Worksheet
Set varFeuille = ActiveSheet
MsgBox varFeuille.Cells.Count
MsgBox ActiveSheet.Cells.Count
End Sub
```



Durant la saisie, vous êtes aidé à plusieurs reprises et en particulier dès que vous saisissez le point après varFeuille, puis après le point de Cells.
Il n'en est pas de même lorsque vous saisissez le point après ActiveSheet. Aucune aide n'est disponible alors qu'il s'agit bien du même objet et que l'exécution du code donne bien le même résultat. Il ne faut donc pas hésiter à utiliser largement les variables-objet. Elles se placent au sein du code comme l'objet lui-même.

9.5 TYPES DE DONNÉES PERSONNALISÉS

Dans la lignée des variables matricielles qui permettent de créer des tableaux de variables de même type, le mot clé Type permet de créer des variables multiples mais de type différents. Cela peut se révéler pratique quand un ensemble de données revient de façon récurrente au sein d'un projet. Il suffit alors de créer ce nouveau Type regroupant les différents types nécessaires. Cette déclaration devra se faire en amont dans la partie déclaration en début de module.
Chaque élément du Type ainsi créé deviendra une propriété du nouveau Type et à ce titre les menus d'aide à la saisie se dérouleront une fois la déclaration effectuée (voir exemple ci-après).

9.5.1 SYNTAXE À UTILISER POUR CRÉE UN NOUVEAU TYPE

L'exemple suivant illustrera la syntaxe. Nous le supposerons inclus dans un projet de gestion d'un club photo.

```
'Définition d'un nouveau "Type"
Type Mb_Club_Photo ' nouveau type Membre du club photo
   Nom As String * 30 ' Nom Membre (max 30 car.)
   Prénom As String * 30 'Prénom Membre
   Inscri_Le As Date ' Date d'inscription
   Adresse As String * 30 ' Adresse membre
```



```
   Ville As String * 30 ' Ville
   CP As String * 5 ' Code postal
   Téléphone As String * 15
   Numérique As Boolean ' Utilise un numérique ?
End Type
Sub Inscrip_Membre()
Dim varNouveau As Mb_Club_Photo '
    With varNouveau
        .Nom = "MARTIN"
        .Prénom = "Isabelle"
        .Inscri_Le = #5/23/2005# ' saisie effectuée
          ' réellement : #23/5/2005#
          ' VBE la valide au format US
        .Adresse = "28 rue d'Aumale"
        .Ville = "Chantilly"
        .CP = "60500"
        .Numérique = True
    End With
' ----- suite du code
End Sub
```

9.6 DÉCLARATION ET PORTÉE DES VARIABLES

9.6.1 INSTRUCTION DIM

Lorsque vous déclarez des variables, vous devez utiliser des instructions de déclaration. Le plus souvent on utilise l'instruction **Dim**.
Ces instructions de déclaration peuvent être placées à l'intérieur d'une procédure ou au début d'un module.
L'instruction suivante crée la variable varNom et lui affecte le type de données String.

```
Dim varNom As String
```

Si cette instruction apparaît à l'intérieur d'une procédure, la variable varNom ne peut être utilisée que dans cette procédure. On dira que sa portée est celle de la macro qui la contient.
En revanche, si cette instruction est située au début du module en dehors de toute macro, la variable varNom sera disponible pour toutes les procédures du module, mais pas pour les autres modules du même classeur. Sa portée sera alors du niveau Module.

Vous pouvez déclarer plusieurs variables dans une même instruction. Si vous souhaitez indiquer un type de données, **vous devez le faire pour chaque variable**.
Dans l'instruction suivante, les variables varB, varC et varD sont déclarées comme étant du type Integer.

```
Dim varB As Integer, varC As Integer, varD As Integer
```

Dans l'exemple suivant, les variables varX et varY ne sont pas déclarées. Par défaut elles seront de type Variant. En fait, seule la variable varZ est déclarée en tant qu'Integer.

```
Dim varX, varY, varZ As Integer
```

9.6.2 INSTRUCTION PUBLIC

L'instruction **Public** est utilisée comme l'instruction **Dim**, à ceci près qu'**elle apparaît en dehors des procédures, au début du module**.

```
Public varNom As String
```

Lorsque vous déclarez une variable comme étant publique, elle peut être utilisée dans toutes les procédures de tous les modules de votre classeur. Elle peut également être utilisée dans n'importe lequel des classeurs qui font référence au classeur dans lequel la variable publique est déclarée. Une telle variable est alors de portée **Projet**.

9.6.3 APPEL DE PROCÉDURES CONTENUES DANS D'AUTRES CLASSEURS

Avant de pouvoir appeler des procédures situées dans un autre classeur, vous devez établir une référence à ce classeur (une connexion entre votre classeur et celui qui contient la procédure que vous voulez appeler).
Pour ce faire :

1) Dans **VBA** cliquez dans le menu **Outils > Références…**
2) Dans la boîte de dialogue **Références-VBA project**, cochez le nom du classeur contenant la procédure que vous voulez appeler. Si le nom du classeur n'apparaît pas dans la liste, cliquez sur le bouton **Parcourir…** pour ajouter le nom dans la liste.

Lorsque la référence a été établie, vous pouvez appeler n'importe quelle procédure publique dans le classeur auquel il est fait référence. Ce classeur apparaît également dans la liste **Bibliothèques de l'Explorateur d'objets** lorsque le classeur possédant la référence est actif.

9.6.4 INSTRUCTION PRIVATE

L'instruction **Private**, de la même manière que l'instruction Public, **apparaît uniquement en dehors des procédures, au début du module**. Lorsque vous utilisez l'instruction Private, la variable qu'elle déclare **ne peut être utilisée que dans les procédures du même module.**

```
Private varNom As String
```

Remarque : L'instruction Dim, lorsqu'elle est utilisée au niveau du module, équivaut à l'instruction Private.

9.6.5 INSTRUCTION STATIC

Avant d'affecter une valeur à une variable déclarée, cette dernière possède une valeur par défaut qui dépend de son type de données. À la fin d'une macro, la valeur des variables est perdue sauf à la déclarer **Static**. Dans ce cas, tant que l'application Excel est active, la valeur de la variable sera conservée en mémoire et sera réutilisable dans un appel ultérieur de la macro.

De plus, suivant qu'elle aura été déclarée Public ou non, cette variable restera disponible pendant toute la cession Excel ou simplement dans l'environnement du fichier contenant la macro.

```
Static varNombre
```

Le mot clé **Static** peut également être mis devant le mot clé **Sub**. Toutes les variables de la macro concernée seront alors affectées de la propriété Static.

```
Static Sub MaMacro()
Dim varNombre As Integer, varNom As String
' varNombre et varNom seront "Static"
```

Résumé :
Lorsque vous déclarez une variable, vous indiquez à une macro comment la nommer. En même temps, vous pouvez indiquer le type de données qu'elle doit contenir. Par exemple, l'instruction suivante crée une variable nommée varRéponse et indique qu'elle est de type Boolean (True ou False).

```
Dim varRéponse As Boolean
```

Vous pouvez rendre disponible une variable soit dans une procédure, un module ou tous les modules d'un même classeur ou tous les modules de tous les classeurs en utilisant les instructions **Dim**, **Public**, **Private** et **Static**

Conseil :
Si vous n'utilisez pas le type de données approprié pour une variable ou un argument auquel un type de données a été affecté, une erreur de type de données peut se produire. Pour éviter de telles erreurs, utilisez uniquement des variables implicites (type de données Variant) sachant que cela entraînera l'utilisation de plus d'espace mémoire.

Exemple 1 :
```
Sub Exemple()
Dim varValeur As Integer
varValeur = 1000
- - - - - -
End Sub
```

Exemple 2 :
```
Sub Calcul_de_coût()
'je ne déclare pas de variable
varA = 0
varB = 20
varC = 30
varD = varA + varB + varC
- - - - - -
End Sub
```

Exemple 3 :
```
Sub EntrezMotPasse()
Dim varNom As String * 5 ' 5 digits max
varNom = "" ' initialisé à "vide"
```

```
MsgBox•">"•&•varNom•&•"<"•' mais varNom contient 5 espaces
varNom•=•InputBox("Entrez•votre•mot•de•passe•(max•5•digits)")
MsgBox•"Votre•mot•de•passe•est•>"•&•varNom•&•"<"
End•Sub•' >< matérialisent les espaces
```

9.6.6 OPTION EXPLICIT

Cette option s'utilise au niveau module pour forcer la déclaration explicite de toutes les variables de ce module.
Si vous utilisez l'instruction **Option Explicit**, **celle-ci doit apparaître dans un module avant toute instruction qui déclare des variables.**
Si vous n'utilisez pas l'instruction Option Explicit, toutes les variables non déclarées possèdent le type de donnée Variant.
Lorsque vous utilisez l'instruction Option Explicit, toutes les variables doivent être déclarées explicitement à l'aide d'une instruction Dim, Private, Public, ReDim ou Static. Si vous tentez d'utiliser un nom de variable non déclarée, une erreur se produit lors de l'interprétation du code dans VBE.

Conseil :
Utilisez l'instruction Option Explicit pour aider à éviter toute faute de frappe lors de la saisie du nom d'une variable existante ou pour écarter tout risque de confusion dans un code où la portée de la variable n'apparaît pas clairement.

Si dans VBA à partir du menu de **Outils > Options…>** onglet **Éditeur**, vous cochez l'option **Déclaration des variables obligatoire**, à chaque nouveau module le libellé Option Explicit est automatiquement générée.

```
Option•Explicit•' Rend obligatoire la déclaration des variables.
Sub•Essai()
Dim•varMonNombre•' Déclare la variable varMonNombre
- - - - -
varMonEntier•=•10•' Ici, cette variable non déclarée
' généra une erreur.
- - - - -
varMonNombre•=•10•' Ne génère pas d'erreur.
- - - - -
End•Sub
```

