Michel Cartereau, mai 2010

AIDE-MÉMOIRE MINIMAL DE VISUAL BASIC POUR EXCEL

Ce document présente sommairement le langage *Visual basic application* dans sa version 6 associée au tableur Excel 2000, afin de fournir un aide-mémoire minimal en cas de nécessité d'utilisation de ce langage.

I - LE LANGAGE VISUAL BASIC

I.1 SYNTAXE GÉNÉRALE

Le langage Visual basic s'écrit sous la forme d'un texte à raison d'une déclaration ou une instruction par ligne a priori, éventuellement poursuivie à la ligne suivante à l'aide de la notation _ ; il est possible de placer plusieurs instructions sur une seule ligne à l'aide de la notation `:`, ainsi que d'intercaler des lignes vides ou de placer un commentaire en fin de ligne ou entre les lignes. Il est sage d'encadrer les symboles (`&`, `:`, etc.) par au-moins une espace. Les noms, définis par Visual basic (mot-clefs, constantes, etc.) ou créés par le programmeur, peuvent s'écrire indifféremment en lettres minuscules ou majuscules (pas de différenciation).

`:`

Symbole introduisant une nouvelle instruction à la suite de l'instruction précédente, sur la même ligne.
Exemple : `compteur = 10 : rang = 1 ' espaces avant et après le deux-point`

`'`

Symbole introduisant un texte de commentaire écrit jusqu'à la fin de la ligne, éventuellement placé après une instruction.
Exemple : `ligne = UCase(ligne)  ' passage en majuscules pour l'enregistrement`

_

Symbole (trait en bas, de souligné) placé en fin de ligne, avec une espace devant, afin d'indiquer la poursuite de l'instruction à la ligne suivante.
Exemple :
```
message = "Bonjour " & nom & _   '  espace avant le trait en bas
 " et bienvenue sur ce vol intergalactique vers Vébéha"
```
Remarque : ce mécanisme ne peut pas être utilisé pour répartir une chaîne de caractères sur plusieurs lignes.

commentaire

Texte libre non pris en compte par Visual basic ; il se note après `'` jusqu'en fin de ligne, ou sur une ligne introduite par le mot-clef `Rem`.

`Rem`

Mot-clef introduisant un texte de commentaire (*remark*) écrit sur la suite de la ligne, avec une espace avant le texte ; ce mot-clef doit figurer en tout début de ligne.
Exemple : `Rem fait le 3 I 2006`

I.2 TYPES

Un type définit la nature d'une variable ou d'un paramètre, qui peut être un type élémentaire : `Boolean`, `Integer`, `Long`, `Byte`, `Single`, `Double`, `String`, `Currency`, `date`, `Object`, `Variant`, `Enum` ou un type composé à l'aide du mot-clef `Type`, ou une classe.

`Boolean`

Ce type correspond à une valeur logique : `True` (vrai) ou `False` (faux).

`Byte`

Ce type correspond à une valeur entière positive codée dans un octet (*byte*) , de 0 à 255.

classe

Une famille d'objets définie dans Visual basic, avec des propriétés et des méthodes associées, qui correspond aussi à un type de même nom.
Exemple : `dim feuille as Worksheet ' une feuille de calcul d'Excel`

`Currency`

Ce type correspond à une valeur monétaire, avec une précision maximale de 15 chiffres en partie entière et de 4 chiffres en partie décimale ; une valeur se note avec le point (« . ») comme séparateur décimal et non pas la virgule (« , »).

Date

Ce type correspond à une date et-ou une heure ; une valeur peut se noter sous la forme universelle #*aaaa*-*mm*-*jj*# pour une date, #*hh*:*mm*:*ss*# pour une heure, ou #*aaaa*-*mm*-*jj hh*:*mm*:*ss*# globalement, avec un recodage par Visual basic en notation anglo-saxonne (mois avant jour, heure le matin « AM » ou l'après-midi « PM ») : #*mm*/*jj*/*aaaa hh*:*mm*:*ss x*M#

Exemple : #2005-5-8 21:43# pour le 8 mai 2005 à 21h43, recodée en #5/8/2005 9:43:00 PM#

Remarques :

- La valeur d'une date est assimilable à un nombre décimal : la quantité de jours écoulés depuis l'origine des dates dans Excel (le 30 décembre 1899 à 0 heure), avec en partie décimale, la fraction de jour correspondante à une heure ; exemple : #2005-5-8 12:00# correspond à 38480,5
- Il est possible de faire un cacul avec une date, pour par exemple passer au jour suivant en l'incrémentant, ainsi que de comparer deux dates pour déterminer par exemple la plus ancienne ; exemples : #2005-5-8 12:00# + 1 correspond au 9 mai 2005 12 heures
  et #2005-11-1# est plus grand que #2005-5-8#
- La valeur nulle est : #1899-12-30 00:00#

Double

Ce type correspond à un grand nombre réel, de valeur absolue comprise entre 10^{-324} et 10^{+308} ; une valeur se note avec le point (« . ») comme séparateur décimal et non pas la virgule (« , »), et avec la lettre « e » ou « E » avant la puissance de 10.

Exemples : 0.48E+101 3.14159265

Enum

Ce type correspond à un ensemble constitué de constantes particulières, et identifié par un nom. Les constantes sont définies explicitement et correspondent à des valeurs entières. Le type doit être défini au niveau global d'un module et non pas à l'intérieur d'une procédure ou d'une fonction.

Squelette de la définition :

```
Enum nom du type
 nom1 = valeur1
 nom2 = valeur2
 ...
End Enum
```

Exemple :

```
Enum typeArbre
 arbreFeuillu = 1
 arbreConifere = 2
End Enum
Dim code As typeArbre
code = arbreConifere
```

Remarques :

- Il est possible de fixer la portée du type au-delà du module en préfixant le mot-clef Enum par Public ou Private ; en l'absence d'indication, la portée est étendue à tout le projet.
- La valeur d'une constante n'est pas obligatoire : en cas d'absence d'indication, c'est alors la valeur zéro dans le cas du premier élément, ou sinon la valeur de l'élément précédent incrémentée de 1.

Integer

Ce type correspond à un petit nombre entier compris entre - 32 727 et 32 728 (2^{15}).

Remarque : la valeur par défaut d'indication est fixée à la valeur nulle (0).

Long

Ce type correspond à un grand nombre entier compris entre - 2 147 483 648 et 2 147 483 647 (2^{31}).

Object

Ce type correspond à la désignation (ou « référence ») d'un objet, obtenue notamment via l'instruction d'affectation Set. L'objet indéfini est désigné par la notation Nothing.

Single

Ce type correspond à un petit nombre réel, de valeur absolue comprise entre 10^{-45} et 10^{+38} ; une valeur se note avec le point (« . ») comme séparateur décimal et non pas la virgule (« , »), et avec la lettre « e » ou « E » avant la puissance de 10.

Exemples : 0.414214E+01 3.14159265

`String`

Ce type correspond à une chaîne de caractères ; la taille d'une valeur est a priori variable (au maximum 64 000 caractères) et elle se note entre guillemets (« " ») ; l'insertion d'un guillemet à l'intérieur d'une chaîne se note par un double guillemet.

Exemples :

```
Dim message As String
Dim nom As String * 40
message = "bonjour ""carotte"""
```

Remarques :

- La comparaison de deux chaînes de caractères s'effectue selon l'ordre « lexicographique », en comparant successivement chaque caractère à partir du premier, selon l'ordre défini par le rang dans l'alphabet ASCII étendu sous Windows ; ainsi `"a"` est supérieure à `"A"`, `"Ali"` supérieure à `"Alain"` et la chaîne vide est la plus petite.
- Il est possible de fixer une taille précise lors de la déclaration à l'aide de la notation : `String *` *taille*
- La valeur nulle est la chaîne vide : `""`
- En Visual basic, une chaîne n'est pas assimilable à un tableau (pas d'indexation au sein de la chaîne).

`Type`

Mot-clef introduisant la définition d'un nouveau type composé de plusieurs éléments (« champs »), déclarés à la suite sous la forme d'une déclaration de variable sur chaque ligne ; le champ d'une variable déclarée avec un type composé est accessible en préfixant le nom de la variable par un point (« `.` ») suivi du nom du champ, ou aussi via l'instruction `With`.

Squelette :

```
Type nom du type
 déclaration d'un champ
 ...
End Type
```

Exemple :

```
Type inscription
 nom As String
 prenom As String
 age As Integer
End Type
Dim etudiant as inscription
etudiant.prenom = "Tarzan"
```

`Variant`

Le type de variable pouvant prendre successivement des valeurs de natures différentes (`Boolean`, `Integer`, `Long`, `Single`, `Double`, `String`, `Currency`, `date`) ; ce type est automatiquement affecté lorsqu'aucune déclaration n'en fixe un. Il est possible de déterminer si une variable de ce type contient ou non une valeur à l'aide de la fonction `IsEmpty()`.

I.3 CONSTANTES ET VARIABLES

Visual basic permet de manipuler des constantes et des variables, simples ou en tableaux, avec des déclarations associées.

`As`

Mot-clef introduisant l'indication du type dans la déclaration d'une constante ou d'une variable, dans la définition d'un paramètre de procédure ou de fonction, dans la définition du résultat d'une fonction.

`Const`

Mot-clef introduisant la déclaration d'une constante.

constante

Une constante est une donnée de valeur fixe (voir à valeur de constante), définie soit automatiquement dans l'environnement de Visual basic, soit explicitement dans un module, une procédure ou une fonction, selon le modèle général : `Const` *nom* `As` *type* = *expression*

Exemples :

```
Const euroFranc As Currency = 6.55957 ' conversion : euro vers franc
vbYesNo
```

Remarques :

- En l'absence de déclaration explicite (`Public`), la portée d'une constante est restreinte au niveau de sa déclaration (celui du module, ou de la procédure ou fonction).
- Les constantes définies dans l'environnement de Visual basic (« intrinsèques ») portent généralement un

nom avec un préfixe particulier (exemple de préfixe : `vb` pour Visual basic).

déclaration

Définition d'une constante (Const), d'une variable (Dim, Private, Public, Static), d'un champ de type composé (`Type`) ou d'un paramètre de fonction ou de procédure ; une définition est composée de :

- L'identification par un nom.
- L'indication facultative d'une organisation en tableau avec la ou les plages d'indiçage notées entre parenthèses.
- L'indication facultative du type associé après le mot-clef `As`.
- L'indication d'une valeur initiale, dans le cas d'une constante (mais pas pour une variable).

Exemples :

```
Const euroFranc As Currency = 6.55957  ' conversion : euro vers franc
Dim nombre As Integer  ' taille de la liste des noms
Public inscrits(1 to 10) as String  ' liste des noms des inscrits
Private nbErreurs As Integer  ' nombre d'erreurs
```

Remarques :

- Une déclaration peut être seule ou figurer dans une liste de déclarations.
- En l'absence d'indication du type, c'est `Variant` qui est automatiquement choisi.
- La valeur initiale d'une variable est la valeur par défaut associée à son type.
- La déclaration d'une variable n'est pas a priori obligatoire car celle-ci est implicitement définie lors de sa première utilisation en l'absence de déclaration, sauf en cas d'utilisation de l'instruction `Option Explicit` ; cependant, il est sage de déclarer systématiquement toutes les variables afin d'éviter notamment la confusion entre une variable de portée limitée à une procédure et une autre homonyme de portée générale.

`Dim`

Mot-clef introduisant une déclaration ou une liste de déclarations de variable, avec une portée limitée au niveau de cette déclaration (c'est-à-dire au module si placée en tête ou sinon dans la fonction ou procédure englobante). Voir aussi `Public` et `Private` pour une déclaration avec une portée explicite.

Exemple : `Dim nombre As Integer  ' taille de la liste des noms`

`Empty`

Constante correspondant au cas où une variable du type `Variant` n'a pas encore de valeur ou à une valeur nulle pour les autres types (sauf objet) : zéro si numérique, chaîne vide si texte, faux si logique.

Remarque : ne pas confondre avec une valeur indéfinie définie par les constantes `Null` et `Nothing`.

`False`

Constante logique (type `Boolean`) valant faux. Lors d'opérations de conversion numérique, la valeur `False` correspond à zéro.

liste de déclarations

Une ou plusieurs déclarations, soit d'un paramètre de fonction ou de procédure, soit d'une variable à la suite des mots-clef `Dim`, `Private` ou `Public`, soit d'une constante à la suite du mot-clef `Const`; deux déclarations successives sont séparées par une virgule (« , »).

Exemples :

```
Dim rang As Integer, taille As Integer
Sub texte(taille As Integer, police As String)
```

Remarques :

- Il est conseillé d'éviter cette liste de déclarations au profit d'une déclaration unique par ligne, pour une meilleure lisibilité.
- Si une des déclarations de la liste ne comporte pas de type, c'est `Variant` qui s'applique et non pas le type éventuellement à la suite ;
  exemple : `Dim rang, taille As Integer ' rang de type Variant et non pas Integer`

`Me`

Objet défini dans une procédure privée, associée généralement à un événement ou un élément d'une boîte de dialogue (contrôle), désignant l'objet à laquelle s'applique la procédure.

Exemple :

```
me.TextBoxNom.Text ' texte de la zone TextBoxNom dans
  ' la boîte de dialogue en cours
```

nom

Mot servant à identifier une constante, variable, une procédure ou une fonction : il débute par une lettre suivie éventuellement d'autres symboles mais pas le point (« `.` »), le trait d'union (« `-` »), le dièse (« `#` »), le et-commercial (« `&` »), le pourcent (« `%` »), l'arobase (« `@` »), le point d'exclamation (« `!` ») ou le dollar (« `$` ») ; il est sage de n'utiliser que des lettres anglaises (sans accent) ou des chiffres.

Exemples : `annee  ligneCourante  option3`

Remarques :

- Le nom peut comporter jusqu'à 255 caractères.
- Il n'y a pas de différenciation des minuscules et des majuscules ; l'éditeur de Visual basic corrige automatiquement une réécriture dans une casse différente.

nombre

valeur numérique correspondant aux types `Integer` (petit entier), `Long` (grand entier), `Byte` (valeur d'octet), `Single` (petit réel) ou `Double` (grand réel).

`Nothing`

Constante désignant un objet indéfini (ou « objet nul »).

Exemple : `Set cellule = Nothing`

Remarques :

- Cette constante s'utilise notamment avec l'opérateur `Is` dans un test afin de déterminer par exemple si une variable de type objet issue d'un traitement correspond bien à un objet défini.
- Ne pas confondre avec la constante `Null` qui s'applique à tout type autre qu'objet.

`Null`

Constante correspondant au cas où une variable, qui n'est pas du type objet, n'a pas une valeur valide.

Remarque : ne pas confondre avec la constante `Nothing` qui s'applique uniquement au type objet ou la constante `Empty` qui correspond à la valeur nulle du type.

portée

La portée indique à quel endroit du projet un élément (une constante, une variable, une fonction ou une procédure) est utilisable ; ce peut être à des niveaux plus ou moins restreints : au niveau général du projet (c'est-à-dire dans tous ses modules), au niveau d'un unique module ou au niveau élémentaire d'une seule procédure ou fonction. En l'absence d'indication explicite, la portée d'une constante ou d'une variable est limitée au niveau de sa déclaration (un module, une procédure ou une fonction), celle d'une procédure ou fonction est générale au projet (tous ses modules). La portée se définit explicitement avec les mots-clef `Public` ou `Private`.

`Private`

Mot-clef indiquant une portée restreinte au module ; il se place soit avant une déclaration ou une liste de déclarations de constante, avant la déclaration d'une procédure ou d'une fonction, soit au début de la déclaration ou d'une liste de déclarations de variable, à la place de `Dim` (en tête de module).

Exemples :

```
Private Sub trace(message As String)
Private nbErreurs As Integer  ' nombre d'erreurs
```

`Public`

Mot-clef indiquant une portée générale au projet ; il se place soit avant la déclaration ou une liste de déclarations de constante, avant la déclaration d'une procédure ou d'une fonction, soit au début de la déclaration ou d'une liste de déclarations de variable, à la place de `Dim` (en tête de module).

Exemples :

```
Public Const taux As Currency = 0.206
Public nomUtilisateur as string
```

tableau

Cas de variable comportant une ou plusieurs plages de valeurs, repérées par un indice selon la ou les dimensions du tableau déclarées de deux manières possibles. Premier cas de déclaration de tableau « fixe » : la ou les plages d'indiçages sont indiquées dans la déclaration de la variable au sein de parenthèses (« `()` ») placées entre le nom et le type, séparées entre elles par une virgule (« `,` »), et notée chacune soit par la taille de la plage, soit par le premier et le dernier indices séparés par le mot-clef `To`. Second cas de déclaration de tableau « dynamique » : aucune indication de plages d'indiçages, notation avec des parenthèses vides ; la ou les plages seront définies plus tard à l'aide de l'instruction `Redim`.

Exemple :
```
Dim inscrits(10) As String  ' a priori, indice de 0 a 10
Dim matrice(1 to 100, 1 to 100) As Double
Dim elus() as String ' dimensions du tableau fixées plus tard
inscrits(0) = "Popeye"
matrice(1, 1) = 1
Redim elus(1 to nombre)
```
Remarques :
- En absence d'indication explicite, la numérotation d'indice débute a priori à zéro, ou à un si fixé via `Option Base` ; il est donc sage d'indiquer explicitement le premier et le dernier indice de chaque plage afin d'éviter toute ambiguïté.
- Il est possible de connaître respectivement le premier ou le dernier indice d'une dimension d'un tableau à l'aide respectivement des fonctions `Lbound(`*tableau*`)` et `Ubound(`*tableau*`)`.
- La fonction `Array(`*liste*`)` permet de créer un tableau à partir d'une liste de valeurs.

`True`
Constante logique (de type Boolean) valant vrai. Lors d'opérations de conversion numérique, la valeur `True` correspond à une valeur non nulle.

valeur de constante
Une constante peut représenter une valeur qui est soit un nombre, soit logique (`False`, `True`), soit une chaîne de caractères, soit une date et-ou une heure, soit indéfinie (`Empty`, `Null`).

valeur par défaut
Il s'agit de la valeur par défaut d'indication automatiquement prise en compte dans le cas de l'utilisation d'une variable ou d'un paramètre non définis, selon le type associé : `False` (faux, type logique), `0` (zéro, type numérique), `""` (chaîne vide), `#1900-01-00 00:00:00#` (origine des dates, type de date), `Empty` (indéfini, type `Variant`).

variable
Elément de mémoire identifié par un nom, avec un type de valeurs possibles et une portée associée ; la déclaration du type d'une variable est soit explicite (à l'aide des mots-clefs `Dim`, `Public` ou `Private`), soit implicite dès sa première utilisation (si non interdit par `Option Explicit`), avec automatiquement le type `Variant`.
Exemples :
```
Dim nombre As Integer  ' taille de la liste des noms
Public inscrits(1 to 10) as String  ' liste des noms des inscrits
Public anomalie As String  ' texte de l'anomalie
```
Remarque : il est sage de déclarer explicitement chaque variable en activant `Option Explicit`.

I.4 PROCÉDURES ET FONCTIONS

argument
Appellation anglaise en Visual basic d'un paramètre d'appel

`ByRef`
Mot-clef à placer devant le nom d'un paramètre d'appel dans la déclaration d'une fonction ou d'une procédure afin d'indiquer un passage de paramètre par référence (la fonction ou procédure accède directement à la variable indiquée lors de l'appel et peut donc modifier la valeur de cette variable originale).
Remarque : c'est le mode de passage activé par défaut d'indication.

`ByVal`
Mot-clef à placer devant le nom d'un paramètre d'appel dans la déclaration d'une fonction ou d'une procédure afin d'indiquer un passage de paramètre par valeur (la fonction ou procédure reçoit une copie de la variable indiquée à l'appel et ne peut donc pas modifier la valeur de cette variable originale).
Remarque : ce n'est pas le mode de passage activé par défaut d'indication (en fait : `ByRef`).

`Call`
Mot-clef pour indiquer l'appel d'une procédure avec indication éventuelle d'une liste de paramètre d'appel.

fonction

Ensemble d'instructions identifié par un nom, avec d'éventuels paramètres associés, qui renvoie un résultat indiqué via la notation : *nom de la fonction* `=` *résultat*. La liste de déclarations de paramètre se notre entre parenthèses (« `()` »), éventuellement vide en cas d'absence de paramètres, suivie du mot-clef `As` et de l'indication du type du résultat. Le passage de paramètre s'effectue a priori par référence en l'absence d'indication particulière (ou avec le mot-clef `ByRef`), sauf indication contraire à l'aide du mot-clef `ByVal` placé devant le nom du paramètre.

Squelette de définition :

`Function` *nom de la fonction*`(`*liste facultative de déclarations de paramètre*`) As` *type du résultat*
instructions avec indication du résultat
`End Function`

L'appel de fonction se note avec son nom suivi de la liste de paramètre d'appel entre parenthèses (« `()` ») ; en cas de fonction sans paramètres, les parenthèses sont en fait facultatives.

Exemple de définition :

```
Function cube(taille As Double) As Double
' renvoie le cube de taille - 28/12/2005 par MC
  cube =  taille ^ 3
End Function
```

et d'appel : `volume = cube(dimension)`

Remarques :

- Il est possible de déclarer un paramètre optionnel en le faisant précéder du mot-clef `Optional`.
- Pour appeler une fonction sans utiliser son résultat, il faut utiliser la notation d'appel de procédure : `Call` *fonction*(*liste de paramètres*)
- En l'absence de déclaration explicite (`Private`), la portée d'une fonction est générale à tout le projet.
- L'exécution d'une fonction peut être abrégée à l'aide de l'instruction : `Exit Function` ; il ne faut cependant pas oublier d'indiquer au préalable son résultat.
- Dans le cas d'une réutilisation de la fonction dans une formule dans Excel (sous la forme d'une fonction personnalisée), il faut veiller à séparer les paramètres d'appel par un point-virgule (« `;` ») et non pas une virgule (« `,` »).

`Function`

Mot-clef introduisant la définition d'une fonction.

`Optional`

Indication d'un paramètre optionnel dans la déclaration d'une fonction ou d'une procédure.

Exemple : `Sub titre(police As String, Optional gras As Boolean = False)`

Remarques :

- Lors de l'appel sans indication de la valeur d'un paramètre, il suffit de ne rien écrire à la position correspondante ou d'utiliser la technique de nommage des paramètres (voir à paramètre d'appel).
- Il est possible, et même sage, d'indiquer une valeur en cas d'absence du paramètre à l'appel (sinon, c'est la valeur par défaut d'indication associée au type) ; ceci se note en faisant suivre la déclaration du paramètre par « `=` » puis une valeur.

paramètre d'appel

Valeur transmise à une procédure ou une fonction lors de son appel, qui peut être une variable, une constante ou une expression.

Exemples :

```
volume = cube(taille)
Call texte(police:="Arial", taille:=12)
```

Remarques :

- Une liste de plusieurs paramètres d'appel se note en les séparant par une virgule (« `,` »).
- Dans le cas d'une liste de paramètres, les valeurs correspondent aux paramètres déclarés dans la définition de la procédure ou de la fonction, selon leur position ; cependant, il est possible de bousculer l'ordre des paramètres ou de ne pas indiquer des paramètres optionnels, en préfixant chaque paramètre d'appel par le nom déclaré suivi de « `:=` » (« paramètres nommés »).
- Le passage de paramètre s'effectue a priori par référence en l'absence d'indication particulière (ou avec le mot-clef `ByRef`), sauf indication contraire à l'aide du mot-clef `ByVal` placé devant le nom du paramètre.
- Le terme anglais correspondant à « paramètre d'appel » est *argument*.

procédure

Ensemble d'instructions identifié par un nom, avec d'éventuels paramètres associés, ne renvoyant pas de résultat ; la liste de déclarations de paramètre se notre entre parenthèses (« `()` »), éventuellement vide en cas d'absence de paramètres. Le passage de paramètre s'effectue a priori par référence en l'absence d'indication particulière (ou avec le mot-clef `ByRef`), sauf indication contraire à l'aide du mot-clef `ByVal` placé devant le nom du paramètre.

Squelette de définition :

```
Sub nom de la procédure (liste facultative de déclarations de paramètre)
  instructions
End Sub
```

L'appel d'une procédure se note soit avec le mot-clef `Call` suivi du nom de la procédure et de la liste de paramètre d'appel entre parenthèses (« ( ) »), soit avec le nom de la procédure suivi de la liste de paramètres sans parenthèses ; en cas de procédure sans paramètres, les parenthèses sont en fait facultatives.

Exemples de définition :

```
Sub texte(taille As Integer, police As String)
' modification de la taille et de la police du texte de la feuille courante
' MC I 2006
  Cells.Select
  Selection.Font.Size = taille
  Selection.Font.Name = police
End Sub
```

et d'appel :

```
Call texte(12, "Arial")
texte 12, "Arial"
Call texte(police:="Arial", taille:=12)
```

Remarques :

- L'exécution d'une procédure peut être abrégée à l'aide de l'instruction : `Exit Sub`
- Il est possible de déclarer un paramètre optionnel en le faisant précéder du mot-clef `Optional`
- En l'absence de déclaration explicite (`Private`), la portée d'une procédure est globale à tout le projet.
- Possibilité de création automatique par le mécanisme d'enregistrement d'une macro (OUTILS MACRO NOUVELLE MACRO).
- Une procédure comportant des paramètres ne peut pas être exécutée de manière interactive.
- La méthode d'un objet est un cas de procédure.

`Static`

Cas particulier de déclaration d'une variable locale à une fonction ou une procédure, dont la valeur est conservée entre chaque appel.

Exemple :

```
Function RangPassage() As Integer
' renvoi du rang d'appel de cette fonction
  Static rang As Integer
  rang = rang + 1
  RangPassage = rang
End Function
```

`Sub`

Mot-clef introduisant la définition d'une procédure (« *subroutine* »).

I.5 OPÉRATIONS

Une opération peut avoir pour résultat une valeur numérique, une valeur logique, une date et-ou une heure ou une chaîne de caractères.

+

Addition de deux valeurs numériques, décalage de dates ou concaténation de deux chaînes de caractères (il est sage d'utiliser plutôt &).

Exemples :

```
8 + 9  ' vaut  17
"Bonjour " + "Chat"  ' vaut  "Bonjour Chat"
#2006-1-21# + 14  ' vaut  #2006-2-4#
```

-

Soustraction de deux valeurs numériques ou différence entre deux dates et-ou heures (résultat décimal exprimé en jours)

Exemples :
```
37 - 5  ' vaut  32
#2006-1-1 13:00# - #2005-12-25 1:00#  ' vaut  7,5
```

*****
Multiplication de deux valeurs numériques ; exemple : `9 * 4 ' vaut 36`

/
Division de deux valeurs numériques ; exemple : `124 / 12  ' vaut 10,3333333333333`

****
Quotient entier de la division de deux valeurs numériques ; exemple : `124 \ 12  ' vaut 10`

^
Elévation à la puissance d'une valeur numérique ; exemple : `10 ^ 3 ' vaut 1000`

&
Opération de concaténation de chaînes de caractères, c'est-à-dire la réunion du texte des deux chaînes.
Exemple : `"Bonjour " & "Chat"` vaut `"Bonjour Chat"`

=
Egalité de deux valeurs, à résultat logique. ; à ne pas confondre avec l'instruction d'affectation (=).
Exemple : `If reponse = "" then reponse = "?" ' initialisation car vide`

<>
Différence de deux valeurs, à résultat logique.
Exemple : `If rang <> maximum then fin = True ' trouve en rang`

<
Infériorité stricte de la valeur de gauche sur la valeur de droite ; comparaison sur valeur numérique, sur date ou sur chaîne de caractères, à résultat logique.
Exemples :
```
If heurechoisie < #9:00# then heurechoisie = #9:00# ' trop tôt !
If reponse < nombre then message = "Trop petit !"
```

<=
Infériorité ou égalité de la valeur de gauche par rapport à la valeur de droite ; comparaison sur valeur numérique, sur date ou sur chaîne de caractères, à résultat logique.
Exemples :
```
If heure <= #21:00# then message = "jour"
If lettre <= "Z" then casse = "majuscule"
```

>
Supériorité stricte de la valeur de gauche sur la valeur de droite ; comparaison sur valeur numérique, sur date ou sur chaîne de caractères, à résultat logique.
Exemples :
```
If heurechoisie > #18:00# then heurechoisie = #18:00# ' trop tard !
If reponse > nombre then message = "Trop grand !"
```

>=
Supériorité ou égalité de la valeur de gauche par rapport à la valeur de droite ; comparaison sur valeur numérique, sur date ou sur chaîne de caractères, à résultat logique.
Exemples :
```
If date >= datedepart then message = "absence"
If lettre >= "a" then casse = "minuscule"
```

And
Conjonction (« et ») de deux valeurs logiques ; résultat vrai si et seulement si les deux opérandes sont vrais.
Exemple : `If (nom <> "") and enmajuscules then nom = UCase(nom)`

Imp
Implication de deux valeurs logiques ; résultat faux si et seulement si le premier opérande est vrai et le second faux.
Exemple : `If terme1 Imp terme2 then terme3 = True`

Is

Egalité entre deux variables désignant un objet ; résultat vrai si et seulement si les deux variables désignent le même objet ; à ne pas confondre avec la comparaison dans une instruction de choix (`Select Case`) ou une condition portant sur le type d'un objet (`If`).

Exemple : `If cellule Is ActiveCell then choix = True`

Like

Comparaison d'une chaîne avec un motif ; résultat vrai si et seulement si la chaîne correspond au motif.

Un motif se note sous la forme d'une chaîne de caractères comportant des symboles génériques :

- `?` : un caractère quelconque (y compris l'espace).
- `*` : aucun, un ou plusieurs caractères quelconques (y compris l'espace).
- `#` : un chiffre unique (de `0` à `9`).
- `[`*liste*`]` : l'un des caractères décrits dans la liste qui est :
  une énumération de caractères et-ou d'intervalles notés $caractère_{début}$-$caractère_{fin}$
- `[!`*liste*`]` : aucun de l'un des caractères décrits dans la liste.
- `[?]` : un véritable point d'interrogation.
- `[*]` : une véritable astérisque.
- `[#]` : un véritable dièse.

Exemples :

```
an like "200#"  ' la notation d'un nombre entier compris entre 2000 et 2009
mot like "[A-Z]*[!.]"  ' tout mot débutant par une lettre majuscule et non terminé par un point.
```

Mod

Reste entier de la division de deux valeurs numériques ; exemple : `124 Mod 12 '` vaut `4`

Not

Négation (« non ») d'une valeur logique ; résultat vrai si et seulement si l'opérande est faux.

Exemple : `If Not trouve then rang = 0`

Or

Disjonction (« ou inclusif ») de deux valeurs logiques ; résultat faux si et seulement si les deux opérandes sont faux.

Exemple : `If (rang < 0) or (rang > 9) then erreur = True`

TypeOf

Condition particulière sur le type d'un objet utilisable uniquement dans l'instruction `If`.

Exemple : `If TypeOf zone Is CommandButton then nature = "bouton"`

Xor

Exclusion (« ou exclusif ») de deux valeurs logiques ; résultat vrai si et seulement si un seul opérande est vrai.

Exemple : `If (pays <> "F") xor (code <> "33") then erreur = True`

I.6 INSTRUCTIONS

=

Instruction d'affectation d'une valeur à une variable ; à ne pas confondre avec l'opération de comparaison par égalité (=) ; il existe une ancienne forme inutilisée désormais (Let).

Exemple : `nom = "carotte"`

Remarques :

- Dans le cas ou la valeur et la variable sont de types différents, une conversion se produit automatiquement et auntant que possible vers le type de la variable.
- La récupération d'une référence vers un objet s'effectue à l'aide de l'instruction `Set`.

Do

Instruction d'itération conditionnelle où la condition est vérifiée :

soit au début de chaque itération (schéma « tantque faire »), « tant que la condition est vérifiée faire » :

```
Do While condition d'itération
  instructions
Loop
```

soit à la fin (schéma « répéter jusqu'à »), « répéter jusqu'à ce que la condition soit vérifiée » :

```
Do
  instructions
Loop Until condition d'arrêt
```

Exemples :

```
Do
  taille = taille * 2
Loop Until taille > 1000
Do while nombre >= 0.001
  nombre = nombre / 2
Loop
```

Remarques :

- Il est possible d'arrêter une itération par l'instruction `Exit Do`.
- En cas d'itération devenue trop longue à l'exécution ou sans fin, arrêter manuellement l'exécution à l'aide des touches ECHAP ou CTRL+ATTN.
- D'autres formes de l'itération existent aussi, basées sur une inversion de la condition :
  a) « tant que la condition n'est pas vérifiée faire » :
  `Do Until` *condition d'arrêt*
  instructions
  `Loop`
  b) « répéter jusqu'à ce que la condition ne soit pas vérifiée » :
  `Do`
  instructions
  `Loop While` *condition d'itération*
  ou sur l'insertion de la condition dans les instructions :
  c) itération avec sortie interne :
  `Do`
  instructions avec sortie conditionnelle (`Exit Do`*)*
  `Loop`

End

C'est à la fois un mot-clef marquant la fin de la définition d'une fonction (`Function`) ou d'une procédure (`Sub`) ou d'un type composé (`Type`), d'une instruction de choix (`If`, `Select Case`) ou d'association (`With`), et à la fois une instruction provoquant l'arrêt immédiat de l'exécution en cours.

Exemple :

```
If nbErreurs > 0 Then
  End
End If
```

Exit

Instruction spécifique à une itération (`Do`, `For`), une procédure ou une fonction, qui abrège son exécution et entraîne la poursuite normale de l'exécution en cours.

Exemples : `Exit Function  Exit Sub  Exit Do  Exit For`

For

Instruction d'itération inconditionnelle, à nombre fixe de répétitions, avec un indice numérique variant d'une valeur initiale à une valeur finale selon un pas fixé a priori à 1 (schéma « pour de à faire »).

Squelette générall :

`Dim` ***indice*** `as Integer`
`For` ***indice*** = ***valeur initiale*** `To` ***valeur finale***
instruction(s)
`Next` ***indice***

Il existe une autre forme adaptée au cas d'un tableau ou d'une collection, itérant sur chaque élément de sa (première) plage d'indiçage :

`Dim` *`element`* `as Variant`
`For Each` *`element`* `In` ***tableau***
instruction(s)
`Next` *`element`*

Exemples :

```
Dim valeur As Double, liste(1 To 10) As Double
Dim compteur As Integer, element As Variant
Randomize
For compteur = 1 To 10
 liste(compteur) = Rnd
Next compteur
For compteur = 10 To 2 Step -1
 liste(compteur) = liste(compteur) + liste(compteur - 1)
Next compteur
```

```
valeur = 0
For Each element In liste
 valeur = valeur + element
Next element
```

Remarques :

- Il n'est pas en fait obligatoire de rappeler le nom de l'indice dans l'instruction `Next`, mais c'est conseillé afin de faciliter la détection d'anomalies dans le cas de répétitions imbriquées.
- Il est possible de fixer une valeur au pas, en rajoutant après la valeur finale : `Step` *valeur du pas*
- Il est très fortement déconseillé de modifier la valeur de l'indice au sein des intructions répétées.
- L'indice peut être de valeur réelle et sa variation peut être décroissante (cas de pas négatif).
- Il est possible d'arrêter l'itération à l'aide de l'instruction `Exit For`

`If`

Instruction liée à une condition, avec la possibilité de prise en compte de l'autre cas où la condition n'est pas satisfaite (`Else`) et-ou de considérer une condition supplémentaire (`Elsif`).

Squelettes :

`If` ***condition*** `then`
 instruction(s)
`End if`

ou aussi avec une alternative quand la condition n'est pas satisfaite :

`If` ***condition*** `then`
 instruction(s)
`Else`
 instruction(s)
`End if`

ou aussi avec plusieurs conditions :

`If` ***condition*** `then`
 instruction(s)
`Elseif` ***condition*** then
 instruction(s)
...
`End if`

ou aussi avec plusieurs conditions et l'alternative quand aucune condition n'est satisfaite :

`If` $condition_1$ `then`
 instruction(s)
`Elseif` $condition_2$ then
 instruction(s)
...
`Else`
 instruction(s)
`End if`

Exemple :

```
If pays = "FR" Then
  prefixe = "0"
Else
  prefixe = "00 33"
End If
```

Remarques :

- Une condition est une expression à résultat de type logique, satisfaite quand elle prend la valeur vraie (`True`) ; l'expression peut être une variable de type logique, une opération simple ou multiple (avec un parenthèsage possible) à résultat logique.
- Il existe une condition spécifique liée au type d'un objet : `TypeOf` ***objet*** `Is` ***type d'objet***.
- Dans le cas où plusieurs conditions sont successivement emboîtées (`If Elseif)`, la première condition satisfaite est choisie, sinon si aucune condition n'est satisfaite, c'est l'éventuelle alternative (`Else)` qui est choisie.
- Il existe une ancienne forme de l'instruction en une seule ligne mais son usage n'est pas conseillé :
  `If` ***condition*** `then` ***instruction*** ou `If` ***condition*** `then` ***instruction*** `else` ***instruction***

`Let`

Ancienne forme de l'instruction d'affectation (=), inutilisée désormais.

Exemple : `Let nom = "Carotte"`

`Loop`

Instruction marquant la fin d'une itération conditionnelle (`Do`).

`Next`

Instruction marquant la fin d'une itération inconditionnelle (`For`).

`Option`

Instruction facultative placée en tout début de module et permettant d'y régler certains paramètres du fonctionnement de Visual basic :

- Obligation de déclaration des variables : `Option Explicit` (automatiquement insérée dans tout nouveau module via la commande OUTILS OPTIONS EDITEUR DÉCLARATION DE VARIABLES OBLIGATOIRES)
- Fixation du premier indice d'un tableau : `Option Base` suivi de `0` ou de `1`
- Fixation du mode de comparaison : `Option Compare` suivi de `Binary` ou `Text` (pas de différence entre minuscule et majuscule de la même lettre)

Exemples :

```
Option Base 1  ' 1er indice de tableau : 1
Option Compare Text  ' comparaison sans distinction de la casse des lettres
```

`Randomize`

Initialisation du générateur de nombres « aléatoires » tirés au hasard, accessibles via la fonction `Rnd`.

`Redim`

Cas d'un tableau déclaré selon le mode « dynamique » (avec `Dim()` : fixation de la ou les plages d'indiçages indiquées au sein de parenthèses (« `()` ») placées entre le nom du tableau et le type de l'élément, séparées entre elles par une virgule (« , »), et notée chacune soit par la taille de la plage, soit par le premier et le dernier indices séparés par le mot-clef `To`.

Exemple :

```
Dim elus() as String ' déclaration initiale du tableau dynamique
Redim elus(1 to nombre) ' fixation de la dimension du tableau
Redim Preserve elus(1 to (UBound(elus) + 1) ) ' un élément supplémentaire
```

Remarques :

- En absence d'indication explicite, la numérotation d'indice débute a priori à zéro, ou à un si fixé via `Option Base` ; il est donc sage d'indiquer explicitement le premier et le dernier indice de chaque plage afin d'éviter toute ambiguïté.
- On peut appliquer plusieurs fois l'instruction `Redim` à un même tableau ; cependant le tableau est réinitialisé à chaque fois sauf si le mot-clef `Preserve` est indiqué après `Redim`.

`Select Case`

Instruction de choix selon les valeurs d'une expression de type numérique ou chaîne de caractères.

Squelette :

`Select Case` *expression à comparer*
`Case` *cas*$_1$
  instruction(s)
`Case` *cas*$_2$
  instruction(s)
...
`Case Else`
  instruction(s)
`End Select`

où un cas peut être décrit comme un cas élémentaire :

- *expression* : le cas est alors activé si la valeur de l'expression à comparer correspond à celle de l'expression.
- *expression*$_1$ `To` *expression*$_2$: activation si la valeur de l'expression à comparer est dans la plage comprise entre *expression*$_1$ et *expression*$_2$
- `Is` *comparaison expression* : activation si la valeur de l'expression à comparer satisfait la comparaison (« `= <> > >= < <=` ») avec *expression*

ou comme un cas multiple :

- liste de cas élémentaires séparés par une virgule (« , ») : activation si la valeur de l'expression à comparer satisfait l'un des cas élémentaires.

Le premier cas activé provoque l'exécution des instructions associées et la poursuite de l'exécution après la fin de l'instruction (`End Select`) ; si aucun cas n'est activé et si le cas complémentaire (`Else`) est présent, celui-ci est alors automatiquement activé. Il est sage de systématiquement ajouter le cas complémentaire.

Exemple :
```
Select Case niveau
Case 0, 1
 etat = "ARRET"
Case Is < 50
 etat = "FAIBLE"
Case 50 To 1000
 etat = "NORMAL"
Case Else
 etat = "FORT"
End Select
```

`Set`

Instruction d'affectation de la référence d'un objet à une variable, qui sert ensuite à désigner cet objet, et qui est a priori du type `Object`.

Squelette : `Set` *variable* = *objet*

Exemple :
```
Dim feuille As Object
Set feuille = Worksheets(2)  ' seconde feuille de calcul
feuille.Activate  ' activation de la feuille de calcul
```

`With`

Instruction facilitant la manipulation d'une variable de type composé (`Type`) en évitant de répéter le nom de la variable à chaque manipulation d'un de ses champs (préfixés cependant par le point, « `.` »).

Squelette :
```
With nom de la variable
  instructions avec accès simplifié aux champs
End With
```

Exemple :
```
Type inscription
 nom As String
 prenom As String
 age As Integer
End Type
Dim etudiant as inscription
With etudiant
 .nom = "De La Jungle"
 .prenom = "Tarzan"
 .age = 22
End With
```

I.7 GESTION D'ERREUR À L'EXÉCUTION

Les instructions présentées ici permettent de gérer la survenue d'une erreur lors de l'exécution du programme écrit en Visual basic.

Err

Objet associé à une erreur d'exécution, comprenant les principales propriétés suivantes :

- `Number` : numéro entier d'identification de l'erreur.
- `Description` : texte explicatif sur la nature de l'erreur.
- `Source` : texte d'identification de l'application à l'origine du déclenchement de l'erreur.

Exemple : `MsgBox "Erreur : " & Err.Description, , "alerte"`

Remarques :

- Cet objet est réinitialisé automatiquement après l'exécution de l'instruction `On Error` ainsi qu'à la sortie d'une procédure (`End Sub`) ou d'une fonction (`End Function`) ; il est aussi possible d'utiliser la méthode `Raise` afin de le réinitialiser par programmation. .
- La propriété `Source` correspond ici dans le cas général soit à l'application Excel (`"Excel.Application"`), soit au programme écrit en Visual basic (`"VBAProject"`).

gestionnaire d'erreur

Le gestionnaire d'erreur prend en charge la survenue d'une erreur lors de l'exécution à partir du moment où il est activé par l'instruction `On Error`. Il se présente sous la forme d'un bloc d'instructions précédé d'un repère d'identification, constitué par un numéro ou une étiquette de ligne. Lorsque l'erreur survient, l'exécution du programme est alors déroutée vers le gestionnaire d'erreur.

Exemple :
```
Sub squelette()
...
On Error GoTo gestion
...
Exit Sub
' traitement d'erreur
gestion:
 MsgBox "Erreur : " & Err.Description, , "alerte"
End Sub
```
Remarques :
- Les numéro ou étiquette de ligne servant à repérer le code d'un gestionnaire se notent sous la forme respective d'un nombre entier (suite de chiffres) ou d'un nom (mot commençant par une lettre), terminés par le symbole deux-points (« : ») et placés au tout début d'une ligne avant la première instruction du gestionnaire.
- Attention à éviter l'exécution du gestionnaire par simple passage en séquence à partir de l'instruction située immédiatement avant sa définition (d'où par exemple l'utilisation de `Exit Sub` juste avant).
- Il est possible d'utiliser l'instruction `Resume` à la fin du gestionnaire afin de reprendre l'exécution au niveau ou juste après l'instruction à l'origine de l'erreur.
- Les informations sur l'erreur sont contenues dans l'objet `Err`.

`On Error`

Instruction associée à la gestion d'erreur lors de l'exécution, utilisée sous l'une des formes suivantes :
- `On Error GoTo` *repère* : activer un gestionnaire repéré par un numéro ou une étiquette de ligne.
- `On Error Resume Next` : désactiver tout traitement d'erreur, l'exécution se poursuivant normalement à la suite en cas d'erreur.
- `On Error GoTo 0` : revenir à la gestion normale par l'application Excel, et annuler toute éventuelle activation précédente d'un gestionnaire particulier.

Exemple : `On Error GoTo gestion`

Remarques :
- Si l'instruction `On Error` est utilisée dans une procédure, le gestionnaire associé doit aussi s'y trouver.
- La gestion normale d'une erreur par Excel consiste en l'affichage d'une boîte d'information sur la nature de l'erreur et l'arrêt brutal de l'exécution du programme.

`Resume`

Instruction réservée au cas d'un gestionnaire d'erreur, qui permet de reprendre l'exécution de plusieurs manières possibles :
- `Resume` : réexécution de l'instruction à l'origine de l'erreur.
- `Resume Next` : reprise de l'exécution juste après l'instruction à l'origine de l'erreur.

Exemple : `Resume Next`

I.8 CAS DES COLLECTIONS

Une collection est ici une classe d'objets correspondant à un ensemble d'éléments éventuellement de types différents, où chaque élément est indexé par son rang (compté à partir de 1) et de manière optionnelle aussi par un libellé (clef) au-choix. Ne pas confondre uen collection avec un tableau.

`Add`

Méthode d'une collection effectuant l'ajout d'un élément (*item*), avec une éventuelle clef (*key*) ; l'ajout s'effectue a priori à la fin sauf s'il y a indication d'une position définie comme avant (*before*) ou après (*after*) tel élément identifié par son index (rang ou clef).

Exemples :
```
sac.Add "choux"
sac.Add item:="chèvre", before:=1  ' ajout au 1er rang
sac.Add item:="Seguin", key:="auteur"  ' ajout à la fin, avec une clef
```
Remarques :
- Il n'y a pas besoin de nommer le paramètre *item* s'il est le seul paramètre d'appel.
- Les paramètres *before* et *after* sont exclusifs : c'est soit l'un, soit l'autre, soit aucun des deux.
- Une erreur d'exécution survient si la clef existe déjà dans la collection.

`Collection`

Classe des collections.

Exemple : `Dim sac As New Collection`

`Count`

Propriété d'une collection indiquant son nombre d'éléments.
Exemple : `taille = sac.Count`
Remarque : cette propriété ne peut pas être modifiée directement ; le nombre d'éléments évolue avec un ajout (`Add`) ou un retrait (`Remove`) d'élément.

`For Each`

Instruction d'itération des éléments d'une collection (schéma « pour chaque élément faire »).
Squelette générall :

```
Dim element as Variant
For Each element In collection
 instruction(s)
Next element
```

Exemple :

```
Dim sac As New Collection
Dim texte As String
Dim truc As Variant
...
For Each truc In sac
 texte = texte & truc & " "
Next truc
```

Remarques :

- Il n'est pas en fait obligatoire de rappeler le nom de l'indice dans l'instruction `Next`, mais c'est conseillé afin de faciliter la détection d'anomalies dans le cas de répétitions imbriquées.
- Il est possible d'arrêter l'itération à l'aide de l'instruction `Exit For`

`Item(`*index*`)`

Méthode d'une collection renvoyant l'élément situé à l'index indiqué en paramètre.
Exemple : `dernier = sac.Item(sac.Count)`
Remarques :

- s'il n'y a pas d'élément correspondant à l'index indiqué (que ce soit un rang ou une libellé), une erreur d'exécution se produit alors.
- Il est possible d'abréger la notation en indiçant directement la collection ; exemple : `sac(sac.Count)`

`Remove`

Méthode d'une collection effectuant le retrait d'un élément par son index (*index*, un rang ou une clef).
Exemples :

```
sac.Remove 1
sac.Remove "auteur"
```

Remarque : une erreur d'exécution survient s'il n'existe pas d'élément dans la collection à l'index indiqué.

II - FONCTIONS GÉNÉRALES

Cette partie présente les principales fonctions utiles, sans aucune exhaustivité.

II.1 CALCULS

`Abs(`*nombre*`)`
Fonction retournant la valeur absolue du nombre donné en paramètre.
Exemple : `Abs(x)`

`Int(`*nombre*`)`
Fonction retournant la partie entière du nombre donné en paramètre ; voir aussi `Fix(nombre)`.
Exemple : `Int(x)`

`Fix(`*nombre*`)`
Fonction retournant la partie entière du nombre donné en paramètre ; voir aussi `Int(nombre)`.
Exemple : `Fix(x)`

`IsNumeric(`*chaîne*`)`
Fonction à résultat logique indiquant si la chaîne donnée en paramètre correspond à un nombre.
Exemples :
```
IsNumeric("13.56") ' renvoie la valeur vraie
IsNumeric("75005 PARIS") ' renvoie la valeur fausse
```

`Round(`*nombre, position*`)`
Fonction retournant la valeur arrondie du nombre donné en paramètre, selon la position indiquée : 0 pour l'unité, 1 et plus en décimale après la virgule, une valeur négative en multiple de 10 avant la virgule.
Exemples :
```
Round(13.56, 1) ' renvoie 13.6
Round(13.56, 0) ' renvoie 14
Round(12345.2, -2) ' renvoie  12300
```
Remarques:
- Si le paramètre de position n'est pas indiqué, c'est la valeur `0` qui est appliquée (arrondi à l'unité).
- Attention ! le comportement de cette fonction en Visual basic est légèrement différent de son équivalent placé directement dans une formule (`Arrondi`) au sein d'une cellule sous Excel, dans le cas où le chiffre à arrondir est 5 : l'arrondi s'effectue alors vers un chiffre pair, ainsi `Round(12,35)` et `Round(12,45)` valent tous deux `12,4`. Il est possible d'utiliser la fonction équivalente dans Excel via la notation suivante : `Application.WorksheetFunction.Round()` avec des paramètres identiques.

`Rnd`
Fonction (sans paramètres a priori) retournant une valeur réelle (type `Single`) tirée au hasard entre 0 (inclus) et 1 (non inclus) ; afin d'obtenir une série aléatoire originale, utiliser au préalable l'instruction `Randomize`.
Exemple :
```
Randomize
rang = Int( (Rnd * 9) + 1 )  'chiffre au hasard entre 1 et 9
```

`Sgn(`*nombre*`)`
Fonction retournant un entier selon le signe de la valeur donnée en paramètre : `1` si positif strictement, `0` si nul, `-1` si négatif strictement.
Exemple :
```
If Sgn(nombre) = -1 Then
  resultat = "Impossible de calculer la racine : nombre négatif !"
Else
  resultat = "Racine carrée de " & nombre & " : " & Sqr(nombre)
End If
```

`Sqr(`*nombre*`)`
Fonction retournant la racine carrée du nombre (positif) donné en paramètre.
Exemple : `Sqr(2)`

II.2 MANIPULATIONS DE TABLEAUX

`Array(`*liste***`)`**

Fonction avec en paramètre plusieurs valeurs, retournant le tableau correspondant, du type `Variant` et dont le premier indice est soit 0, soit 1 selon le réglage courant via `Option base`.

Exemple :

```
Dim liste As Variant
liste = Array("choux", "poux", "cailloux")
```

`IsArray(`*variable***`)`**

Fonction avec en paramètre le nom d'une variable, retournant la valeur vraie si le contenu de cette variable correspond à un tableau, déclaré avec `Dim` ou créé via `IsArray()`.

Exemple :

```
Dim liste As Variant
If not IsArray(liste) then
 liste = array("?") ' tableau à 1 élément
End If
```

`Lbound(`*tableau***`)`**

Fonction avec en paramètre le nom d'un tableau et le rang d'une de ses dimensions, retournant le premier (*lower*) indice de cette dimension du tableau : `lbound(`***tableau, dimension***`)` ; si le rang est omis, c'est la première dimension qui est considérée.

Exemples : `Lbound(inscrits)` `Lbound(matrice, 2)`

`Ubound(`*tableau***`)`**

Fonction avec en paramètre le nom d'un tableau et le rang d'une de ses dimensions, retournant le dernier (*upper*) indice de cette dimension du tableau : `lbound(tableau, dimension)` ; si le rang est omis, c'est la première qui est considérée.

Exemples : `Ubound(inscrits)` `Ubound(matrice, 2)`

II.3 MANIPULATIONS DE TEXTE

`Asc(`*string***`)`**

Fonction avec en paramètre une chaîne réduite à un caractère, retournant le rang correspondant dans le code ASCII

Exemple : `Asc("~")` `'` renvoie la valeur 126

`Chr(`*charcode***`)`**

Fonction avec en paramètre un rang dans le code ASCII, retournant le caractère (*character*) correspondant.

Exemple : `chr(34)` `' guillemet : "`

Remarque : le retour à la ligne dans une cellule s'obtient avec `chr(10)`.

`CStr(`*number***`)`**

Fonction de conversion en chaîne, renvoyant le texte correspondant à la valeur donnée en paramètre.

Exemple : `CStr(Date)` `' date sous forme abrégée, exemple : "11/05/2010"`

Remarque : une valeur logique est convertie dans le texte `True` ou `False`.

`Format(`*expression, format***`)`**

Fonction de mise en forme, renvoyant un texte correspondant à la valeur donnée en paramètre (nombre ou date) présentée selon les indications du paramètre de format dont voici les principales caractéristiques :

- Le format est une chaîne de caractères dont certains symboles (« méta-symboles ») correspondent à un des éléments de la valeur donnée en paramètre et sont remplacés par le texte issu de la mise en forme.
- Cas du jour : numéro avec (« `dd` ») ou sans (« `d` ») zéro devant un seul chiffre, nom abrégé (« `ddd` ») ou complet (« `dddd` »).
- Cas du mois : numéro avec (« `mm` ») ou sans (« `m` ») zéro devant un seul chiffre, nom abrégé (« `mmm` ») ou complet (« `mmmm` »).
- Cas de l'année : décennie avec (« `yy`) » ou sans (« `y` ») zéro devant un seul chiffre, complète (« `yyyy` » ou « `yyy` »).
- Cas de l'heure : numéro avec (« `hh` ») ou sans (« `h` ») zéro devant un seul chiffre.
- Cas des minutes : numéro avec (« `nn` ») ou sans (« `n`) » zéro devant un seul chiffre, ou bien "« `m` »" et "« `mm` »" si immédiatement placés après « `h` » ou "« `hh` »
- Cas des secondes : numéro avec (« `ss` ») ou sans (« `s` ») zéro devant un seul chiffre.
- Cas d'un nombre : position pour un chiffre avec (« `0` ») ou sans (« `#` ») remplacement par zéro si non significatif, indication de partie décimale (« `.` ») avec arrondi automatique de celle-ci selon le nombre de

positions, séparation des milliers (« , »), en notation scientifique avec (« e+ ») ou (« e- ») sans signe en cas d'exposant positif.
- Utiliser la barre inversée (« \ ») pour préfixer dans le format un méta-symbole à afficher tel que sans obtenir la valeur corrrespondante ; exemple : "\h" pour un véritable "h" et non la valeur de l'heure.

Exemples :
```
Format(1125.658, "0,0.00")  ' "1 125,66"
Format(Now, "d mmmm yyyy, hh \h nn")  ' exemple : "21 mai 2010, 11 h 07"
```
Remarques :
- Si le paramètre de format est omis, une présentation est automatiquement appliquée : un nombre tel que, une date ou une heure selon les conventions fixées localement (exemple : "21/05/2010, 11:07:32").
- Il existe aussi des formats abrégés comme par exemple "c" pour date et heure sous forme complète.
- Attention aux méta-symboles « m » et « mm » : normalement ils désignent le numéro du mois, mais s'ils sont placés immédiatement après « h » ou « hh » ils correspondent alors aux minutes.

InStr(*start, string1, string2, compare***)**

Fonction de recherche avec en paramètre le rang de départ (*start*), une chaîne de caractères à examiner (*string1*), une autre chaîne de caractères pour le motif (*string2*) à rechercher au sein de la précédente, et un indicateur (*compare*) de respect de la casse des lettres (vbBinaryCompare ou 0 - différence, vbTextCompare ou 1 - égalité, d'une minuscule et d'une majuscule de la même lettre), retournant le rang de la première apparition du motif dans la chaîne si trouvé, ou 0 si absent.

Exemples :
```
rang = InStr(1, "Hêtre (fagus sylvatica)", "Fagus", 1)  ' vaut 8
rang = InStr(1, "Hêtre (fagus sylvatica)", "Fagus", 0)  ' vaut 0
rang = InStr(13, "Hêtre (fagus sylvatica)", "a", 1)  ' vaut 18
```
Remarques :
- Si le paramètre de la casse n'est pas indiqué, la valeur fixée par Option Compare est prise en compte.
- Le rang du départ est facultatif si celui de la casse est absent, et la recherche démarre alors au 1er rang.

LCase(*string***)**

Fonction avec en paramètre une chaîne de caractères, retournant une copie transformée en lettres minuscules (*lowercase*).

Exemple : `nom = LCase(nom)`

Remarque : le passage en lettres majuscules s'obtient par `UCase(string)`.

Left(*string, length***)**

Fonction avec en paramètre une chaîne de caractères et une taille, retournant la sous-chaîne composée des *taille* premiers caractères.

Exemple :
```
Dim nom As String, id As String
nom = "Popeye"
id = left(nom, 3) ' la valeur "Pop"
```
Remarques :
- Si la chaîne contient moins de caractères que demandé à l'appel, toute la chaîne est renvoyée.
- Si la taille indiquée est nulle, la fonction renvoie une chaîne vide ("").

Len(*string***)**

Fonction avec en paramètre une chaîne de caractères, retournant son nombre de caractères.

Exemple : `taille = Len(nom)`

Mid(*string, start, length***)**

Fonction avec en paramètre une chaîne de caractères, un rang (compté à partir de 1) et une taille, retournant la sous-chaîne composée de taille caractères à partir du rang indiqués.

Exemple :
```
Dim codess As String, moisnaissance As String
codess = "2680575006001"
moisnaissance = Mid(codess, 4, 2) ' la valeur "05"
```
Remarques :
- Si la taille n'est pas indiquée à l'appel, tous les caractères jusqu'en fin de la chaîne sont extraits.
- Si le rang indiqué n'est pas valide, la fonction renvoie une chaîne vide ("").

`Right(`*string, length***`)`**

Fonction avec en paramètre une chaîne de caractères et une taille, retournant la sous-chaîne composée des *taille* derniers caractères.

Exemple :

```
Dim nom As String, id As String
nom = "Popeye"
id = right(nom, 3)  ' vaut la valeur "eye"
```

Remarques :

- Si la chaîne contient moins de caractères que demandé à l'appel, toute la chaîne est renvoyée.
- Si la taille indiquée est nulle, la fonction renvoie une chaîne vide (`""`).

`Str(`*number***`)`**

Fonction avec en paramètre une valeur numérique, retournant une chaîne de caractères la représentant avec une espace devant si positif ou le tiret (« - ») si négatif.

Exemple : `texte = Str(9)  ' vaut la valeur " 9"`

`Trim(`*string***`)`**

Fonction avec en paramètre une chaîne de caractères, retournant une copie transformée par élimination de tout espace situé au début ou à la fin.

Exemple : `nom = Trim(saisie)`

`UCase(`*string***`)`**

Fonction avec en paramètre une chaîne de caractères, retournant une copie en majuscules (*uppercase*).

Exemple : `nom = UCase(nom)`

Remarque : le passage en lettres minuscules s'obtient par `LCase(string)`.

`Val(`*string***`)`**

Fonction avec en paramètre une chaîne de caractères représentant un nombre, retournant la valeur numérique reconnue selon le type Double ; si la chaîne contient un caractère qui ne correspond pas à la notation d'un nombre, la reconnaissance ne prend en compte que les caractères situés avant.

Exemples :

```
val("2006") '  vaut  2006
val("8/1/2006") '  vaut 8
val("0,1") '  vaut 0
```

II.4 MANIPULATIONS DE DATE

`CDate(`*expression***`)`**

Fonction renvoyant l'objet de date correspondant à la chaîne donnée en paramètre.

Exemple : `CDate("12/5/2006") ' vaut #2006-5-12#`

Remarque : si la chaîne en paramètre ne correspond pas à une date, une erreur d'exécution survient alors.

`Date`

Fonction sans paramètre retournant la date courante, sous la forme d'une valeur du type `date`

Exemple : `dateCourante = date`

`DateValue(`*date***`)`**

Fonction avec en paramètre une chaîne contenant une date notée selon la langue du poste, retournant la valeur correspondant de type date.

Exemple : `DateValue("8/1/2006") '  vaut  #2006-1-8#`

`Day(`*date***`)`**

Fonction avec en paramètre une date, retournant le numéro du jour de cette date.

Exemple : `Day("#2006-1-3#") ' vaut 3`

`Hour(`*time***`)`**

Fonction avec en paramètre une date, retournant le numéro de l'heure de cette date.

Exemple : `Hour("#2006-1-3 17:54:30#") ' vaut 17`

`IsDate(`*expression***`)`**

Fonction à résultat logique indiquant si la chaîne donnée en paramètre correspond à une date.

Exemples :

```
IsDate("21/12") ' renvoie la valeur vraie
IsDate("1 mai 2001") ' renvoie la valeur vraie
IsDate("14 VII 2005") ' renvoie la valeur fausse
```

`Minute(`*time*`)`**
Fonction avec en paramètre une date, retournant le numéro de minute de cette date.
Exemple : `Minute("#2006-1-3 17:54:30#") ' vaut 54`

`Month(`*date*`)`**
Fonction avec en paramètre une date, retournant le numéro du mois de cette date.
Exemple : `Month("#2006-1-3 17:54:30#") ' vaut 1`

`MonthName(`*month*`)`**
Fonction avec en paramètre un numéro de mois, retournant le nom du mois (selon la langue du poste). Un second paramètre optionnel permet d'obtenir la forme abrégée du mois si indiqué avec une valeur vraie.
Exemples :
- `MonthName(9) ' vaut "septembre"`
- `MonthName(9, true) ' vaut "sept"`

`Now`
Fonction sans paramètre retournant la date et l'heure courantes, sous la forme d'une valeur du type `date`
Exemple : `horodate = Format(Now, "c")`

`Second(`*time*`)`**
Fonction avec en paramètre une date, retournant le numéro de seconde de cette date.
Exemple : `Second("#2006-1-3 17:54:30#") ' vaut 30`

`Time`
Fonction sans paramètre retournant l'heure courante, sous la forme d'une valeur du type `date`
Exemple : `heureCourante = time`

`Year(`*date*`)`**
Fonction avec en paramètre une date, retournant l'année de cette date.
Exemple : `Year("#2006-1-3#") ' vaut 2006`

II.5 DIVERS

`IsEmpty(`*expression*`)`**
Fonction avec en paramètre une variable (type `Variant`), retournant une valeur vraie si cette variable ne contient pas de valeur.
Exemple :
```
If isEmpty(nom) Then
  nom = "?" ' initialisation car vide
End If
```

`TypeName(`*varname*`)`**
Fonction avec en paramètre une expression, retournant une chaîne avec le nom de son type.
Exemples :
```
TypeName("8/1/2006") ' vaut "String"
TypeName(#2006-1-8#) ' vaut "Date"
```

III - VISUAL BASIC ET EXCEL

III.1 GÉNÉRALITÉS

Visual basic fournit la possibilité de programmer des actions sur les feuilles de calcul d'un classeur d'Excel enregistré dans un projet, par le biais de modules définissant notamment un ensemble de macros ou de fonctions personnalisées ; il permet aussi la construction de boîtes de dialogue avec la programmation des actions associées.

boîte de dialogue
Petit fenêtre destinée notamment à afficher un message, poser une question ou saisir des informations, construite par assemblage de zones et de procédures écrites en Visual basic affectées aux manipulations associées ; l'ensemble de code correspondant est lié à la définition de la boîte de dialogue et le tout peut être sauvegardé dans un fichier à l'extension « `.frm` », en sélectionnant cette définition puis en activant la commande FICHIER EXPORTER UN FICHIER dans l'éditeur de Visual basic.

fonction personnalisée
Une fonction définie en Visual basic pour un projet donné, où elle est utilisable dans une formule d'Excel, notamment via la commande INSERTION FONCTION, dans la catégorie PERSONNALISÉES.

macro
Une procédure sans paramètres définie en Visual basic pour un projet donné ; le déclenchement d'une macro s'effectue dans Excel principalement via la commande OUTILS MACRO MACROS ou, si définis, via un bouton ou un raccourci au clavier.

module
Ensemble de code écrit en Visual basic, rattaché à un projet d'Excel et correspondant essentiellement :
- Soit à un module standard, fournissant notamment des macros et des fonctions personnalisées.
- Soit à un module de feuille, contenant notamment les procédures associés à des événements spécifiques à une feuille de calcul du projet.
- Soit aux actions définies sur un boîte de dialogue.

module de feuille
Ensemble de code écrit en Visual basic, regroupant généralement des procédures associées à des événements et lié à une feuille d'un projet d'Excel ; il peut être sauvegardé dans un fichier à l'extension « `.cls` », en le sélectionnant puis en activant la commande FICHIER EXPORTER UN FICHIER dans l'éditeur de Visual basic.

module standard
Ensemble de code écrit en Visual basic, regroupant des déclarations de constantes et de variables de niveau général, des procédures (dont des macros) ou des fonctions (notamment personnalisées) ; un module est identifié par un nom et rattaché à un projet dans Excel. Il peut être sauvegardé dans un fichier à l'extension « `.bas` », en le sélectionnant puis en activant la commande FICHIER EXPORTER UN FICHIER dans l'éditeur de Visual basic.
Squelette d'un module standard :
réglages avec Option
déclarations de constantes et de variables
procédures et fonctions
Remarque : l'ancienne appellation d'un module standard est « module de code ».

projet
Ensemble regroupant les feuilles de calcul d'un classeur d'Excel avec un ou plusieurs modules écrits en Visual basic ; chaque projet est enregistré dans un fichier unique, à l'extension « `.xls` ».

`UserForm`
Appellation en Visual basic d'une boîte de dialogue.

III.2 PRINCIPAUX OBJETS LIÉS À EXCEL

Cette partie présente les principaux objets permettant de manipuler en Visual basic des éléments d'un classeur dans l'application Excel, résumés dans le tableau suivant :

NIVEAU	OBJETS	CLASSE
Excel	`Application`	`Application`

Classeur	`Workbooks("`*nom*`")`
	`ThisWorkbook`
	`ActiveWorkbook`
		`Workbook`

Feuille	`Sheets` `ActiveSheet`	`Sheet`
Feuille de calcul	`Worksheets`	`Worksheet`
Ligne	`Rows(`*numéro de ligne*`)` `Rows`	`Range`
Colonne	`Columns(`*numéro de colonne*`)` `Columns`	`Range`

Plage de cellules	`Range(`*référence en mode A1*`)`
	`Selection`
	`Cells`
		`Range`

Cellule	`Cells(`*numéro de ligne, numéro de colonne*`)` `ActiveCell`	`Range`
Boîte de dialogue	`UserForms`	`UserForm`
Zone de boîte de dialogue	`Controls`	`Control`

`ActiveCell`

Propriété (de l'application) représentant la cellule active en cours.
Exemple : `ActiveCell.value = "?"`

`ActiveSheet`

Propriété d'un classeur, ou si non indiqué du classeur actif, représentant la feuille de calcul active en cours.
Exemple : `nom = ActiveSheet.name ' nom de la feuille active`

`ActiveWorkbook`

Propriété (de l'application) représentant le classeur actif en cours.
Exemple : `nom = ActiveWorkbook.name ' nom du classeur actif`

`Application`

Objet désignant l'application dans laquelle s'exécute Visual basic, c'est-à-dire Excel dans ce cas présenté ici.
Exemple : `versionExcel = Application.version ' version d'Excel utilisée`

`Cells`

Propriété représentant la collection de toutes les cellules, soit pour une feuille de calcul, soit pour une plage de cellules, soit si non précisé pour la feuille active (`ActiveSheet`) ; dans la collection, une cellule est repérée soit par ses numéros de ligne et de colonne, soit par son rang dans la liste obtenue en parcourant successivement les lignes de gauche à droite, puis du haut vers le bas.
Exemples :

```
Worksheets(1).Cells(2, 1) ' cellule "A2" de la 1ère feuille de calcul
Cells(5, 6) ' cellule "F5" de la feuille de calcul active
Range("C3:E10").Cells(1, 1) ' 1ère cellule de la plage indiquée (C3)
Cells(256) ' cellule "A2" de la feuille de calcul active
ActiveSheet.Cells ' les cellules de la feuille de calcul active
```

`Column`

Propriété représentant le numéro de la première colonne d'une plage de cellules.
Exemple : `Range("J1:M1").Column ' vaut 10`
Remarque : le numéro de la dernière colonne peut s'obtenir avec l'expression suivante :
plage`.Columns(`*plage*`.Columns.Count).Column`

Columns

Propriété représentant la collection de toutes les colonnes de cellules, soit pour une feuille de calcul, soit pour une plage de cellules, soit si non précisé pour la feuille active (`ActiveSheet`).

Exemples :

```
Worksheets(1).Columns(2) ' 2ème colonne de la 1ère feuille de calcul
Columns(3) ' 3ème colonne de la  feuille de calcul active
Range("A1:J1").Columns ' les 10 colonne de la plage indiquée
ActiveSheet.Columns ' les colonnes de la feuille de calcul active
```

Controls

Propriété représentant la collection de toutes les zones d'une boîte de dialogue.

Exemple : `UserForms(1).Controls(1) ' 1ère zone de la 1ère boîte de dialogue`

EnableEvents

Propriété de l'objet de l'application (Excel) à valeur logique contrôlant le déclenchement des événements.

Exemple :

```
Application.EnableEvents = false  ' désactivation des événements
WorkSheets(1).Activate  ' activation de la 1ère feuille sans événement induit
Application.EnableEvents = true  ' réactivation des événements
```

EntireColumn

Propriété représentant toute la colonne d'une cellule ou d'une plage sur une seule colonne.

Exemple : `ActiveCell.EntireColumn`

EntireRow

Propriété représentant toute la ligne d'une cellule ou d'une plage sur une seule ligne.

Exemple : `ActiveCell.EntireRow`

Offset(*RowOffset, ColumnOffset***)**

Propriété représentant une sous-plage obtenue par décalage, relativement à la plage de cellules considérée ; le décalage s'indique en nombre de lignes (*RowOffset*) et en nombre de colonnes (*ColumnOffset*).

Exemple : `ActiveCell.offset(1, 0)  ' cellule en-dessous de celle active`

Range(*sous-plage***)**

Propriété représentant une sous-plage de cellules, soit pour une feuille de calcul, soit relativement à la plage de cellules considérée, soit si non précisé dans la feuille active (`ActiveSheet`) ; la sous-plage de données se note soit sous la forme d'une référence en style « A1 » (avec « : » pour indiquer un bloc), soit sous la forme du couple de références de la cellule en haut à gauche et de celle en bas à droite notées en style « A1 » ou à l'aide de `Cells()`, soit sous la forme d'un nom affecté à cette zone dans Excel.

Exemples :

```
Worksheets(1).Range("A1") ' 1ère cellule de la 1ère feuille de calcul
Range("A1:A10") ' les 10 premières cellules de la 1ère colonne
Range("A1", "A10") ' idem que ci-dessus
Range(Cells(1, 1), Cells(10, 1)) ' idem que ci-dessus
Range("tauxTVA") ' la cellule de nom "tauxTVA"
```

Remarque : il est possible d'agréger plusieurs plages avec la méthode `Union`.

Row

Propriété représentant le numéro de la première ligne d'une plage de cellules.

Exemple : `Range("A10:A15").Row ' vaut 10`

Rows

Propriété représentant la collection de toutes les lignes de cellules, soit pour une feuille de calcul, soit pour une plage de cellules, soit si non précisé pour la feuille active (`ActiveSheet`).

Exemples :

```
Worksheets(1).Rows(2) ' 2ème ligne de la 1ère feuille de calcul
Rows(3) ' 3ème ligne de la  feuille de calcul active
Range("A1:A10").Rows' les 10 lignes de la plage indiquée
ActiveSheet.Rows ' les lignes de la feuille de calcul active
```

Selection

Propriété (de l'application) représentant l'objet sélectionné dans la fenêtre active de l'application, c'est-à-dire généralement la plage de cellules sélectionnées dans la feuille active.

Exemple : `Selection.clear ' vide les cellules de la sélection courante`

Sheets

Propriété représentant la collection de toutes les feuilles (de calcul ou de graphique), soit pour un classeur, soit si non précisé pour le classeur actif (`ActiveWorkbook`) ; dans la collection, une feuille est repérée soit par son rang (à partir de 1), soit par son nom.

Exemples :

```
Sheets(1)) ' 1ère feuille (de calcul ou de graphique)
Sheets("donnees")  ' feuille intitulée "donnees"
```

ThisWorkbook

Propriété (de l'application) représentant le classeur dans lequel s'exécute le code de Visual basic ; généralement, il correspond au classeur actif (`ActiveWorkbook`).

Exemple : `ThisWorkbook.name ' le nom du classeur du code en exécution`

Union(*plage*$_1$, *plage*$_2$)

Méthode (de l'application) fusionnant deux plages (ou plus) de cellules en une seule.

Exemple : `Set paquet = Application.Union(Range("A2:A9"), Range("C2:C9"))`

Remarque : il est possible de rajouter d'autres plages en paramètre pour une fusion de 3 plages ou plus.

UserForms

Propriété représentant la collection de toutes les boîtes de dialogue chargées dans l'application.

Exemple : `UserForms(1) ' 1ère boîte de dialogue chargée`

Workbooks

Propriété représentant la collection de tous les classeurs ouverts ; dans la collection, un classeur est repéré soit par son rang (à partir de 1), soit par son nom.

Exemples :

```
Workbooks(1) ' 1er classeur
Workbooks("parcelle.xls") ' classeur du fichier "parcelle.xls"
```

WorksheetFunction

Propriété (de l'application) permettant d'utiliser une fonction définie dans Excel ; appel via la notation : `WorksheetFunction.`*nom de la fonction*(*paramètres d'appel*)

Exemples :

```
WorksheetFunction.Pi() ' constante trigonométrique Pi
WorksheetFunction.Max("A1:A11") ' valeur maximale de la plage
```

Worksheets

Propriété représentant la collection de toutes les feuilles de calcul, soit pour un classeur, soit si non précisé pour le classeur actif (`ActiveWorkbook`) ; dans la collection, une feuille est repérée soit par son rang (à partir de 1), soit par son nom.

Exemples :

```
Worksheets(1)) ' 1ère feuille de calcul
Worksheets("donnees") ' feuille de calcul intitulée "donnees"
```

III.3 PRINCIPALES MANIPULATIONS DE CELLULE

Voici les principales propriétés et méthodes associées à une cellule, voire à une plage de cellules. A noter que la propriété d'une cellule par défaut d'indication est sa valeur (`Value`).

Borders

Ensemble des bordures d'une cellule identifiées par une constante, pour l'essentiel : `xlEdgeBottom`, `xlEdgeLeft`, `xlEdgeRight`, `xlEdgeTop` ; une bordure a comme propriétés essentielles :

- La couleur : `Color` ou `ColorIndex`
- Le type de trait : `LineStyle` (`xlContinuous`, `xlDash`, `xlDashDot`, `xlDashDotDot`, `xlDot`, `xlDouble`, `xlSlantDashDot` ou `xlLineStyleNone`)
- L'épaisseur du trait : `Weight` (`xlHairline`, `xlThin`, `xlMedium` ou `xlThick`)

Exemples :

```
ActiveCell.Borders(xlEdgeRight).LineStyle = xlContinuous
ActiveCell.Borders(xlEdgeRight).Weight = xlThick
```

Clear

Méthode de cellule, supprimant le contenu et la mise en forme.

Exemple : `ActiveCell.clear`

Color

Propriété (Borders, Font, Interior) correspondant à une couleur, généralement exprimée comme un mélange des 3 couleurs primaires (rouge, vert, bleu) avec une intensité de chacune codée entre 0 (absence) et 255 (maximum) à l'aide de la notation RGB(*rouge*, *vert*, *bleu*)

Exemple : `ActiveCell.Interior.Color = RGB(255, 0, 0) ' fond rouge`

ColorIndex

Propriété (Borders, Font, Interior) correspondant à une couleur, désignée par un rang dans une palette de 56 couleurs (1 noir, 2 blanc, 3 rouge, 4 vert, 5 bleu, 6 jaune etc.)

Exemple : `ActiveCell.Font.ColorIndex = 5 ' texte bleu`

Delete(*décalage***)**

Méthode de cellule, pour effectuer une suppression avec décalage des cellules voisines soit à droite (xlShiftToLeft), soit en-dessous (xlShiftUp), soit si non indiqué choisies par Excel. .

`Exemple : Range("H1:H2").Delete(xlShiftUp)`

Find(*What, After, LookIn, LookAt, SearchOrder, SearchDirection, MatchCase***)**

Méthode de plage de cellules, pour y rechercher une valeur (*What*) ; la recherche s'effectue à partir de la cellule située juste après celle indiquée avec le paramètre *After* mais de manière circulaire dans la plage (retour à la première cellule après la dernière) ; le paramètre *LookIn* indique la nature du contenu de cellule à considérer, soit la valeur (xlValues), soit la formule (xlFormulas), soit le commentaire associé (xlComments) ; la mise en correspondance s'effectue selon le paramètre *LookAt* soit sur l'ensemble du contenu de cellule (xlWhole) soit toute partie de ce contenu (xlPart) ; le mode de parcours de la plage est défini selon le paramètre *SearchOrder* soit par colonne (xlByColumns) soit par lignes (xlByRows) ; le sens de la recherche est défini par le paramètre *SearchDirection* soit en avant (xlNext) soit en arrière (xlPrevious) ; la recherche de texte est réalisée selon le paramètre *MatchCase* en distinguant (True) ou en confondant (False) la minuscule et la majuscule des lettres. La méthode renvoie la référence de la première cellule trouvée dans la plage ou un objet indéfini si la recherche est infructueuse (cf. Nothing).

Exemple :
```
Set cellule = Range("A1:F27").Find(What:="bon", _
 After:=Range("A27"), LookIn:=xlValues, lookAt:=xlWhole, _
 SearchDirection:=xlNext, searchOrder:=xlByRows, MatchCase:=False)
```

Remarques :

- En cas de non indication de la valeur du paramètre *After*, la recherche démarre à partir de la cellule située immédiatement après celle en début de la plage dans le coin supérieur gauche.
- La recherche peut être poursuivie avec les méthodes FindNext() ou FindPrevious().
- Il est sage d'indiquer la valeur des paramètres *LookIn, LookAt, SearchOrder, SearchDirection* et *MatchCase*, car sinon leur valeur est alors celle définie lors de la dernière indication à l'appel des méthode Replace() et Find(), ou bien lors de la dernière utilisation des commandes Edition Rechercher / Remplacer dans le tableur Excel.
- Voir aussi l'opérateur Like notamment pour l'utilisation de motifs de recherche.
- Il existe un paramètre d'appel supplémentaire et final (*MatchByte*) à n'utiliser que dans le cas particulier où Excel est paramétré pour l'usage d'alphabets spéciaux (codage d'un caractère sur deux octets).

FindNext(*After***)**

Méthode de plage de cellules, pour y reprendre une recherche initiée par la méthode Find() ; la recherche s'effectue à partir de la cellule située juste après celle indiquée avec le paramètre *After* mais de manière circulaire dans la plage (retour à la première cellule après la dernière). La méthode renvoie la référence de la première cellule trouvée dans la plage ou un objet indéfini si la recherche est infructueuse (cf. Nothing).

Exemple : `Set cellule = Range("A1:F27").FindNext(cellule)`

Remarque : s'il n'existe qu'une seule occurence de la valeur recherchée, le résultat de cette méthode renverra alors cette occurence (même si trouvée précédemment).

FindPrevious(*After***)**

Méthode de plage de cellules, pour y reprendre en sens inverse une recherche initiée par la méthode Find() ; la recherche s'effectue à partir de la cellule située juste avant celle indiquée avec le paramètre *After* mais de manière circulaire dans la plage (retour à la dernière cellule après la première). La méthode renvoie la référence de la première cellule trouvée dans la plage ou un objet indéfini si la recherche est infructueuse (cf. Nothing).

Exemple : `Set cellule = Range("A1:F27").FindPrevious(cellule)`

Remarque : s'il n'existe qu'une seule occurence de la valeur recherchée, le résultat de cette méthode renverra alors cette occurence (même si trouvée précédemment).

Font

Ensemble des caractéristiques du texte de la cellule : `Name` (police de caractères), `Size` (taille), `Color ou ColorIndex` (couleur), `Bold` (en gras), `Italic` (en italiques), `Underline` (souligné)

Exemple :

```
With Selection.Font ' pour la sélection courante
  .Name = "Arial" ' police Arial
  .Size = 14 ' taille de 14 points
  .Bold = True ' en gras
  .Italic = False ' pas d'italiques
  .Color = RGB(255, 0, 0) ' rouge
End With
```

Formula

Propriété correspondant au contenu de la cellule sous la forme d'une formule en style « A1 », exprimée dans la langue de Visual basic (anglais a priori).

Exemples :

```
Range("A2").Formula = "=A1"
Range("A2").Formula = "=today()"  'date du jour en anglais
```

FormulaLocal

Propriété correspondant au contenu de la cellule sous la forme d'une formule en style « A1 », exprimée dans la langue de Excel sur le poste (français a priori).

Exemples :

```
Range("A2").FormulaLocal = "=A1"
Range("A2").FormulaLocal = "=aujourdhui()"  ' date du jour en français
```

FormulaR1C1

Propriété correspondant au contenu de la cellule sous la forme d'une formule en style « L1C1 », exprimée dans la langue de Visual basic (anglais a priori, avec « `R` » pour la ligne et des crochets « `[ ]` » pour la notation relative).

Exemples :

```
Range("A2").FormulaR1C1 = "=R[-1]C"  ' valeur de la cellule en A1
Range("A2").FormulaR1C1 = "=today()"  'date du jour en anglais
```

FormulaR1C1Local

Propriété correspondant au contenu de la cellule sous la forme d'une formule en style « L1C1 », exprimée dans la langue de Excel sur le poste (français a priori).

Exemples :

```
Range("A2").FormulaR1C1Local = "=L(-1)C"  ' valeur de la cellule en A1
Range("A2").FormulaR1C1Local = "=aujourdhui()"  ' date du jour en français
```

Insert(*décalage*)

Méthode de cellule, pour effectuer une insertion avec décalage de cellules soit vers la droite (`xlShiftToRight`), soit vers le bas (`xlShiftDown`), soit si non indiqué choisi par Excel.

`Exemple : Range("A5:H5").Insert(xlShiftDown)`

Interior

Propriété du fond de la cellule, avec la possibilité de manipuler sa couleur : `Color` ou `ColorIndex`

Exemple : `ActiveCell.Interior.ColorIndex = 3 ' fond rouge`

Replace(*What, Replacement, LookAt, SearchOrder, MatchCase*)

Méthode de plage de cellules, pour remplacer dans le contenu de cellule (valeur ou formule) toute occurence d'un texte original (*What)* par un autre (*Replacement*) ; la mise en correspondance s'effectue selon le paramètre *LookAt* soit sur l'ensemble du contenu de cellule (`xlWhole`) soit toute partie de ce contenu (`xlPart`) ; le mode de parcours de la plage est défini selon le paramètre *SearchOrder* soit par colonne (`xlByColumns`) soit par lignes (`xlByRows`) ; la recherche du texte original est réalisée selon le paramètre *MatchCase* en distinguant (`True`) ou en confondant (`False`) la minuscule et la majuscule des lettres.

Exemple :

```
Selection.Replace What:="bon", Replacement:="bien", _
 lookAt:=xlWhole, searchOrder:=xlByRows, MatchCase:=False
```

Remarques :

- Il est sage d'indiquer la valeur des paramètres *LookAt, SearchOrder* et *MatchCase*, car sinon leur valeur est alors celle définie lors de la dernière indication à l'appel des méthode `Replace()` et `Find()`, ou bien lors de la dernière utilisation des commandes EDITION RECHERCHER / REMPLACER dans le tableur Excel.
- Cette méthode renvoie toujours la valeur vraie (`True`).

- Il existe un paramètre d'appel supplémentaire et final (*MatchByte*) à n'utiliser que dans le cas particulier où Excel est paramètré pour l'usage d'alphabets spéciaux (codage d'un caractère sur deux octets).

`Select`
Méthode de cellule, pour effectuer une sélection.
Exemple : `Range("A1:A10").Select`

`Value`
Propriété correspondant au contenu de la cellule renvoyée sous la forme de sa valeur (et non pas une éventuelle formule) ; en modification, comme pour `Formula`, il est possible d'indiquer une formule en style « A1 », exprimée dans la langue de Visual basic (anglais a priori).
Exemples :
```
Selection.value = 0  ' mise à zéro des cellules sélectionnées
Range("A1").value = "=today()"  ' date du jour dans la 1ère cellule
Range("A1").value ' vaut alors par exemple "05/01/2006"
```
Remarques :
- Utiliser la fonction `IsEmpty()` afin de détecter une cellule à contenu vide.
- Cette propriété est la valeur par défaut d'indication pour une cellule ; ainsi `ActiveCell` et `ActiveCell.value` désignent la même valeur dans une expression.

III.4 PRINCIPALES MANIPULATIONS DE FEUILLES DE CALCUL

Voici les principales propriétés et méthodes associées à une feuille de calcul.

`Activate`
Méthode d'activation d'une feuille existante.
Exemple : `Worksheets(2).Activate`

`Delete`
Méthode de suppression d'une feuille, avec demande interactive de confirmation par Excel ; il est possible d'éviter cette demande de confirmation, en modifiant `Application.DisplayAlerts`.
Exemple :
```
Application.DisplayAlerts = False ' desactivation de la confirmation
Worksheets("transit").Delete
Application.DisplayAlerts = True ' réactivation de la confirmation
```

`Index`
Propriété d'une feuille correspondant à son rang (compté à partir de 1).
Exemple : `message = "Feuille n°" & ActiveSheet.index`

`Name`
Propriété d'une feuille correspondant à son nom.
Exemple : `message = "Feuille " & ActiveSheet.name`

`Protect`
Méthode de protection d'une feuille, avec un mot de passe facultatif.
Exemples :
```
Worksheets("mesures").protect
Worksheets("calcul").protect("Zut!") ' avec mot de passe "Zut!"
```

`Unprotect`
Méthode d'annulation de la protection d'une feuille, avec un éventuel mot de passe.
Exemples :
```
Worksheets("mesures").unprotect
Worksheets("calcul").unprotect("Zut!") ' avec mot de passe "Zut!"
```

`UsedRange`
Propriété d'une feuille : plus petite plage rectangulaire contenant toutes ses cellules utilisées, c'est-à-dire avec une valeur ou une mise en forme définie explicitement.
Exemple : `Activesheet.UsedRange.select`

III.5 QUELQUES MANIPULATIONS DE BOÎTE DE DIALOGUE

Voici une présentation très succincte des boîtes de dialogue, soit utilisées pour afficher un message ou poser une question, soit créées par le programmeur (`UserForm`) ; ces dernières sont généralement désignées par une variable portant leur nom ou, dans une procédure privée associée à un événement lié à une de leurs zones, par le mot-clef `Me`.

`AddItem` *Item*
Méthode d'une zone à liste (`ComboBox`, `ListBox`), ajoutant un texte (*Item*) en dernier élément de la liste.
Exemple : `CBListe.AddItem "Hiboux"`

`Clear`
Méthode d'une zone de saisie (`TextBox`, `ComboBox`, `ListBox`), provoquant l'effacement de son contenu, y compris la suppression de la liste des valeurs possibles dans une zone à liste.
Exemple : `CBListe.clear`

`ComboBox`
Classe de zone où la valeur (`Text`) peut être soit choisie dans sa liste, soit saisie directement ; la liste peut être construite avec `AddItem` ou détruite avec `Clear`.

`Hide`
Méthode de masquage d'une boîte de dialogue.
Exemple : `boiteSaisie.Hide  ' boîte de nom "boiteSaisie"`

`InputBox(`*`Prompt, Title, Default`***`)`**
Fonction d'ouverture d'une boîte de dialogue pour saisie une valeur, avec une question et une valeur proposée, retournant la valeur saisie par l'utilisateur ; principaux paramètres d'appel :
- *Prompt* : texte de la question à afficher dans la boîte au-dessus de la zone de saisie.
- *Title* : facultatif, titre de la boîte de dialogue ; choisi automatiquement par l'application si absent à l'appel.
- *Default* : facultatif, valeur automatiquement proposée ; rien si absent à l'appel.

Exemples :
```
nb = InputBox("Quantité ?", "Saisie", 1)
age = InputBox(Prompt:="Age" ?, Title:="Saisie", Default:=23)
```
Remarques :
- Si l'utilisateur annule la saisie, il est renvoyé une chaîne vide.
- D'autres paramètres permettent de fixer la position de la boîte sur l'écran (*Left*, *Top*) ou de fixer le type de la donnée renvoyée (*Type*) qui est `String` par défaut d'indication.

`ListBox`
Classe de zone où la valeur (propriété `Text`) peut être choisie dans une liste mais pas saisie (pour cela utiliser ComboBox) ; la liste peut être construite avec la méthode `AddItem` ou détruite avec la méthode `Clear`.

`ListCount`
Propriété d'une zone à liste (`ComboBox`, `ListBox`), donnant la taille de la liste de ses valeurs.
Exemple : `nbOptions = CBListe.listCount`

`ListIndex`
Propriété d'une zone à liste (`ComboBox`, `ListBox`), correspondant au rang (compté à partir de zéro) de la sélection courante dans la liste de ses valeurs ; la modification cette propriété provoque la sélection de la valeur de l'élément correspondant.
Exemples :
```
CBListe.ListIndex = 0  ' choix du premier élément
CBListe.ListIndex = CBListe.ListCount - 1  ' choix du dernier élément
```

`MsgBox(`*`Prompt, Buttons, Title`***`)`**
Fonction d'ouverture d'une boîte de dialogue pour afficher le message, avec le titre et l'aspect indiqués, et retournant un code entier selon l'action de l'utilisateur ; principaux paramètres d'appel :
- *Prompt* : texte du message à afficher dans la boîte.
- *Buttons* : code facultatif pour l'aspect correspondant essentiellement soit à un bouton unique de validation (`vbOkOnly`), soit aux boutons de validation et d'annulation (`vbOKCancel`), soit aux boutons de réponse par oui ou non (`vbYesNo`) ; bouton de validation seul si paramètre absent à l'appel.
- *Title* : facultatif, titre de la boîte de dialogue ; choisi automatiquement par l'application si absent à l'appel.

et principaux codes retournées : `vbOK` (validation), `vbCancel` (annulation), `vbYes` (oui), `vbNo` (non).

Exemples :
```
call MsgBox("Rien ne va plus", ,"Alerte")
reponse = MsgBox(Prompt:="Encore ?", Buttons:=vbYesNo, Title:="Calcul")
```

Show
Méthode d'affichage d'une boîte de dialogue, initialement avec création ou après masquage (`Hide`).
Exemple : `boiteSaisie.show  ' boîte de nom "boiteSaisie"`

Text
Propriété d'une zone (`TextBox`, `ComboBox`, `ListBox`), donnant le texte contenu ou sélectionné.
Exemple : `nom = TextNom.Text  ' contenu de la zone de texte "TextNom"`

TextBox
Classe de zone correspondant à la saisie d'un texte (`Text`).

Unload(*boîte*)
Procédure de fermeture d'une boîte de dialogue.
Exemple : `Call Unload(me) ' dans une procédure privée associée à une boîte`

Value
Propriété d'une zone (`TextBox`, `ComboBox`, `ListBox`), donnant le texte contenu ou sélectionné.
Exemple : `nom = TextNom.value  ' contenu de la zone de texte "TextNom"`

zone
Elément (`Control`) d'une boîte de dialogue (`UserForm`) pouvant être une zone de texte (`TextBox`) ou une zone de liste (`ComboBox`, `ListBox`) dans le cadre de cette présentation.

IV - QUELQUES ÉVÉNEMENTS

Activate
Evénement associé à l'activation d'une feuille ou d'un classeur.

AfterUpdate
Evénement associé à la fin de la modification d'une zone de saisie (`TextBox`, `ComboBox`, `ListBox`) ; utile notamment pour une zone de texte (`TextBox`) où il se déclenche seulement à la fin de la saisie ou des modifications, et non pas à chaque frappe ou correction d'une lettre (comme `Change`)

BeforeDoubleClick
Evénement associé au double clic de souris dans une feuille, avec identification de la cellule la plus proche (paramètre `Target`).

BeforeRightClick
Evénement associé au clic avec le bouton droit de la souris dans une feuille, avec identification de la cellule la plus proche (paramètre `Target`).

Calculate
Evénement associé au recalcul d'une feuille.

Change
Evénement associé à la modification du contenu d'une cellule (paramètre `Target`) ou de celui d'une zone de saisie (`TextBox`, `ComboBox`, `ListBox`) ; cet événement ne correspond pas au recalcul de la feuille ou à la suppression de cellule. Dans le cas d'une zone de texte (`TextBox`), il se déclenche à chaque frappe ou correction d'une lettre et non pas uniquement à la fin de la saisie ou des modifications (comme `AfterUpdate`).

Click
Evénement associé au clic sur un bouton de commande, ou à une sélection dans une zone de liste (`ComboBox`, `ListBox`).

Deactivate
Evénement associé à la fin de l'activation d'une feuille ou d'un classeur.

SelectionChange
Evénement associé à une nouvelle sélection (le paramètre `Target` désigne la plage sélectionnée)

V - ALPHABET ASCII ÉTENDU DE WINDOWS

32	*espace*	64	**@**	96	**`**	160		192	**À**	224	**à**
33	**!**	65	**A**	97	**a**	161	**¡**	193	**Á**	225	**á**
34	**"**	66	**B**	98	**b**	162	**¢**	194	**Â**	226	**â**
35	**#**	67	**C**	99	**c**	163	**£**	195	**Ã**	227	**ã**
36	**$**	68	**D**	100	**d**	164	**¤**	196	**Ä**	228	**ä**
37	**%**	69	**E**	101	**e**	165	**¥**	197	**Å**	229	**å**
38	**&**	70	**F**	102	**f**	166	**¦**	198	**Æ**	230	**æ**
39	**'**	71	**G**	103	**g**	167	**§**	199	**Ç**	231	**ç**
40	**(**	72	**H**	104	**h**	168	**¨**	200	**È**	232	**è**
41	**)**	73	**I**	105	**i**	169	**©**	201	**É**	233	**é**
42	*****	74	**J**	106	**j**	170	**ª**	202	**Ê**	234	**ê**
43	**+**	75	**K**	107	**k**	171	**«**	203	**Ë**	235	**ë**
44	**,**	76	**L**	108	**l**	172	**¬**	204	**Ì**	236	**ì**
45	**-**	77	**M**	109	**m**	173		205	**Í**	237	**í**
46	**.**	78	**N**	110	**n**	174	**®**	206	**Î**	238	**î**
47	**/**	79	**O**	111	**o**	175	**¯**	207	**Ï**	239	**ï**
48	**0**	80	**P**	112	**p**	176	**°**	208	**Đ**	240	**ð**
49	**1**	81	**Q**	113	**q**	177	**±**	209	**Ñ**	241	**ñ**
50	**2**	82	**R**	114	**r**	178	**²**	210	**Ò**	242	**ò**
51	**3**	83	**S**	115	**s**	179	**³**	211	**Ó**	243	**ó**
52	**4**	84	**T**	116	**t**	180	**´**	212	**Ô**	244	**ô**
53	**5**	85	**U**	117	**u**	181	**µ**	213	**Õ**	245	**õ**
54	**6**	86	**V**	118	**v**	182	**¶**	214	**Ö**	246	**ö**
55	**7**	87	**W**	119	**w**	183	**·**	215	**×**	247	**÷**
56	**8**	88	**X**	120	**x**	184	**¸**	216	**Ø**	248	**ø**
57	**9**	89	**Y**	121	**y**	185	**¹**	217	**Ù**	249	**ù**
58	**:**	90	**Z**	122	**z**	186	**º**	218	**Ú**	250	**ú**
59	**;**	91	**[**	123	**{**	187	**»**	219	**Û**	251	**û**
60	**<**	92	****	124	**|**	188	**¼**	220	**Ü**	252	**ü**
61	**=**	93	**]**	125	**}**	189	**½**	221	**Ý**	253	**ý**
62	**>**	94	**^**	126	**~**	190	**¾**	222	**Þ**	254	**þ**
63	**?**	95	**_**	127	*DEL*	191	**¿**	223	**ß**	255	**ÿ**

VI - AUTEUR, LICENCE D'USAGE ET VERSION

Il s'agit de la troisième version de cet aide-mémoire, réalisée en mai 2010. Ce document est diffusé pour un usage individuel. Il est librement téléchargeable sur le site de l'auteur. Toutes les remarques et corrections sont les bienvenues. Pour contacter l'auteur :

Michel CARTEREAU
AgroParisTech - UFR d'informatique - 16, rue Claude Bernard - F 75231 PARIS CEDEX 5
http://www.agroparistech.fr/mmip/mc/ - michel.cartereau à agroparistech.fr

TABLE DES MATIÈRES