Visual Basic pour Applications

D. Mailliet

I. Introduction

(Ce document est une présentation de VBA, il s'adresse à des informaticiens et non à des débutants comme le sont les étudiants de SM1)

VBA est un langage de programmation orienté objet interprété qui remplace les versions antérieures des macros des applications de MicroSoft (principalement Acces, Excel, Powerpoint et Word c'est à dire l'intégré Office). Les anciennes macro restent utilisables et mélangeables (autre module) dans un même classeur.

Les mots du langage peuvent s'exprimer (selon un choix initial) soit en anglais soit en français: Ils ne peuvent pas être mélangés à l'intérieur d'un même classeur. Lors de la création du classeur le choix français ou anglais doit avoir été fait.

II. Création d'un Module

Les feuilles de texte de programmes en VBA Excel sont appelées Modules.

1. Lancer **Excel**.
2. 🖱 sur **Outils** / **Options** / 🗀 **Module** / **Paramètres par défaut** / **Anglais;Personnalisé** [1]/ 🗀 **Général** / **Nombre de feuilles par classeur : *3*** / **Ok** . De façon à avoir les mots clés du langage en anglais
3. 🖱 sur **Affichage/Barre d'outil /Visual Basic** / **Ok**. Pour ajouter une barre d'outil permettant de tester les procédures. On peut aussi enlever la devant **Standard** et **Format** pour libérer de la place.
4. Faire glisser la barre Visual Basic sous la barre de titre a l'aide de la.
5. 🖱 du bouton droit sur un 🗀 onglet inférieur (**feuil<i>**) Puis du bouton gauche sur **Insérer** et sur **Module.**
6. Supprimer les éventuelles feuilles inutiles en 🖱 du bouton droit sur un 🗀 onglet inférieur (**feuil<i>**) Puis du bouton gauche sur **Supprimer**. On peut aussi **Renommer** les feuilles restantes
7. On peut aussi déplacer les feuilles les unes par rapport aux autres en 🖱 et les faisant glisser.

(1) si on choisit Français;France, toutes les instructions devront être tapées en français. Anglais;personnalisé permet de garder les dates à la française. Observons néanmoins les séparateurs de listes qui doivent être des « ; ». Certains exemples ont étés écrits avec l'option anglais/Us où le séparateur est une « , ».

III. Type, Variables et constantes

A. Types de base

Type	Nombre d'Octets	Domaine
Boolean	2	True / False
Integer	2	-32768 .. 32767
Long	4	-2147483648.. 2147483647
Single	4	-3,402823E38..-1,401298E-45 ; 1,401298E-45.. 3,402823E38

Double	8	4,94065645841247E-324 .. 1,79769313486232E308 (>0 et <0)
Currency	8	-922337203685477,5808.. 922337203685477,5807
Date	8	1/1/100 à 0:0:0 .. 31/12/9999 à 23:59:59
String	1 par caractère	longueur indéfinie(<2000000) ou fixée (<65536)
Variant	16	prend selon les divers affectations les valeurs précédentes ainsi que Empty Error ou Null
Object	selon l'objet	Objets définis: ex :Range , worksheet , toolbar , etc.. Objets indéfinis: Object
Personnel	somme des tailles des champs	combinaison des types précédents
Array	taille * nbre	Tableaux d'éléments de même nature ou liste d'éléments de toute nature

B. Variables

1. Déclaration explicite ou implicite

Par défaut, en VBA, les déclarations sont implicites : lors de leurs affectations une déclaration implicite des variables est effectuée automatiquement par l'interpréteur ; le type de cette variable est systématiquement : **Variant** .

On peut s'obliger à déclarer explicitement les variables en indiquant **Option explicit** : contrôle orthographique et gain de place et de rapidité.

2. Déclarations explicites

après avoir choisi l' **Option explicit** toute variable doit être déclarée par l'une des formes:

- **Dim** nombre **As** *Integer*
- **Private** nombre **As** *Integer*
- **Public** nombre **As** *Integer*
- **Static** nombre **As** *Integer*

(nombre et *Integer* sont évidemment a remplacer par un nom de variable et un type approprié)

3. Attention

Dim nombre1,nombre2 **As** *Integer* : nombre1 n'est pas déclaré comme **intéger** mais comme **Variant** .

4. Portée - Durée de vie

a) Portée (des variables) globale au classeur

Déclarées en tête de module (après les « **Option** ») et avant les procédures et fonctions par :

Public <variable> **As** <type>

la <variable> est accessible dans tous les modules du classeur

b) Portée (des variables) globale au module

Déclarées en tête de module (après les « **Option** ») et avant les procédures et fonctions par :

Private <variable> **As** <type> *ou* **Dim** <variable> **As** <type>

la <variable> est accessible dans le modules courant mais non dans les autres modules

c) Portée (des variables) locale aux procédures et fonctions

Déclarées en tête de la procédure (ou la fonction) par :

Private <variable> **As** <type> *ou* **Dim** <variable> **As** <type>

la <variable> n'est accessible que dans la procédure (ou la fonction) courante mais non dans les autres procédures (ou fonctions) du module

d) Portée (des procédures et fonctions) globale au classeur

une déclaration :

Sub <ma_proc>**(**<arguments>**)** ...**End sub**

rend la procédure <ma_proc> accessible par toutes les procédures des modules du classeur , par **Outils** / **Macro** / <**maproc**> / **Exécuter**, par un bouton créé par l'utilisateur ou par un menu

e) Portée (des procédures et fonctions) locale au module

Private Sub <ma_proc>**(**<arguments>**)** .. **End sub**

rend la procédure <ma_proc> accessible uniquement par les procédures du module courant, n'est pas visible dans la liste lors **Outils** / **Macro**

f) durée de vie des variables

les variables locales définies par **Dim** ou **Private** sont réinitialisées à chaque appel de procédures, alors que celles déclarées par **Static** conservent leurs valeurs précédentes

C. Constantes

1. Constantes prédéfinies

elles commencent par:

- xl (ex : xlWorksheet) pour les objet Excel
- vb (ex : vbYesNo) pour les objets Visual Basic

on en obtient une liste en sur **Affichage** /**Explorateur d'objet** / **Bibliothéque;classeur** : *Excel* / **Objet;Module** : *Constants* (ou *VBA* au lieu de *Excel*)

2. Constantes utilisateur typées

Const Pi **as integer =** 3.14

3. Constantes utilisateur non typées

Const Pi = 3.14

prend plus de place en mémoire que la constante précédente

D. Exemples

fichier **tst_Type.xls**

```
Option Explicit ' la déclaration des variables est obligatoire
Option Base 1 ' les indices des tableaux commencent à 1 par défaut
```

```
Const sp As String * 1 = " "
Dim nl As String * 1


Type t_perso
  nom As String * 24
  prenom As String * 36
End Type

Dim booleen As Boolean
Dim entier As Integer
Dim ent_Long As Long
Dim re_simp As Single
Dim re_dbl As Double
Dim nb_ech As Currency
Dim dte As Date
Dim chn As String
Dim var As Variant
Dim objet As Object
Dim perso As t_perso
Dim tab1(3) As t_perso
Dim tab2(-5 To 3) As Integer
Dim tab3 As Variant
Dim tab4(3, 2) As Integer

Sub tst_type()
  nl = Chr(10)
  Let entier = 5 ' let est facultatif
  booleen = True
  chn = "salut"
  ent_Long = 12345678
'  ent_Long = "xxxxx"  provoque une erreur : "type incompatible"
  re_simp = 12345678
  re_dbl = 12345678
  nb_ech = 12345678
  dte = 123456
  chn = entier & sp & chn & sp & booleen & nl
'  chn = "53"  :  entier = chn  ' Erreur : types incompatibles
  chn = chn & ent_Long & sp & re_simp & nl
  chn = chn & re_dbl & sp & nb_ech & nl
  chn = chn & dte & nl
  var = 1234
  chn = chn & var & sp
  var = "bonjour"        ' ne provoque pas d'erreur : type variant
  chn = chn & var & nl
  Set objet = Toolbars
'  chn = chn & objet & nl ' ERREUR : toolbars ne possede pas la propriete value
  With perso
    .nom = "effelavesselle"
    .prenom = "vladimir"
  End With
  chn = chn & perso.prenom & sp & perso.nom & nl
  MsgBox chn
  chn = TypeName(booleen) & sp & TypeName(chn) & nl
  chn = chn & TypeName(ent_Long) & sp & TypeName(re_simp) & nl
  chn = chn & TypeName(re_dbl) & sp & TypeName(nb_ech) & nl
  chn = chn & TypeName(dte) & sp & TypeName(chn) & nl
```

```
  var = 1234
  chn = chn & TypeName(var) & sp
  var = "bonjour"
  chn = chn & TypeName(var) & nl
  chn = chn & TypeName(objet) & nl
  MsgBox chn, vbInformation, "Les Types"
  Call MsgBox(chn, vbExclamation, "Les Types") ' autre formulation
'  MsgBox (TypeName(perso)) ' ERREUR : type d'argument byref incompatible
  tab1(2) = perso
  tab2(-4) = -65
  tab3 = Array("bonjour", 3, True)
'   tab2 = Array(1, 2)  ' Erreur: affectation a un tableau impossible
'  tab2 = tab2     ' Erreur : la même
'  tab3 = tab2  ' ne pose aucun problème
  MsgBox tab3(1) & nl & tab1(2).nom & nln & tab3(2) & nl _
  & tab2(-4) & nl & tab3(3) & nl & tab4(1, 1) & sp & tab4(3, 2)
End Sub
```

IV. Affectations

1. différents types d'affectation

- **let** a = 5 — dans le cas de types simples
- a = 5 — ou plus simplement
- **set** obj = MenuBars(''monMenu'') — pour des Objects
- argum **:=** valeur — Pour des arguments nommés (cf procedures)

2. Initialisation automatique

Dès leur déclaration , les variables sont initialisées à 0 pour les nombres et à (''') chaîne vide pour les string et à Nothing pour les objects.

V. Règles d'écriture

classique ou un peu moins

A. Ecrire les mots réservés en minuscule

S'ils sont reconnus, les initiales seront transformées en majuscule dès validation de la ligne (leur liste en français et en anglais se trouve dans le fichier : **ListeVBA.xls**)

B. Syntaxe

En cas d'erreur, un message d'erreur apparaît et la ligne est écrite en rouge

C. Indentation

No comment

D. Commentaires

ils sont précédés de :

- **Rem**

- ‘

VI. Séquences

- 1 instruction par ligne - 1 ligne par instruction
- on peut cependant écrire plusieurs instructions sur une même ligne en les séparant par « : »
- on peut cependant écrire une instructions sur plusieurs lignes en les liant par « _ »

VII. Fonctions et procédures

A. Déclaration de Fonctions

Function <ma_fonc> **(** <arguments> **As** <type arg.>**)** **As** <Type du résultat>

<déclaration des variables locales et statiques>

<Corps de fonction>

<ma_fonc> **=** <résultat>

End Function

As <type arg.> et **As** <Type du résultat> sont facultatifs

B. Déclaration de Procédures

Sub <ma_proc>**(**<arguments> **As** <type arg.> **)**

<déclaration des variables locales et statiques>

<Corps de procédures>

End sub

As <type arg.> est facultatif

C. Arguments

- Le passage d’argument se fait exclusivement par **ADRESSE**
- Le type est facultatif et donc **Variant** par défaut
- On peut déclarer des arguments optionnels (**Optional**) Ils ne doivent pas être typée et doivent être les derniers de la liste des arguments

D. Appel

1. de fonctions

classique :

<ma_var> = <mafonc>(<mes_arg>)

ou pour une fonction qui produit des effets de bords dont on ne voudrai pas récupérer le résultat

<mafonc>(mes_arg>)

exemple classique : MsgBox est une fonction qui renvoie le numéro du bouton sur lequel l’utilisateur a appuyé

(il peut y en avoir de 1 à 3) , s'il y a qu'un bouton il est inutile de mémoriser le résultat:

MsgBox (''<Message>**'')**

ou avec 3 boutons:

<var_reponse> = **MsgBox (''**<Message>**'',VbAbortRetryIgnore)**

2. De procédure

On peut appeler une procédure de 3 façons différentes:

- **Call** <ma_proc> **(** <mes_val>**)**
- <ma_proc> <mes_val>
- <ma_proc> <nom_arg> **:=** <mes_val>

exemple: **MsgBox ''**<Message>'' est ici considéré comme une procédure

E. Exemple

fichier **foncproc.xls**

```
Option Explicit
Dim c As Integer   ' variable globale

Private Sub test_proc(a As String, b As Integer)
  MsgBox b, , a
End Sub

Function dble(x As Integer) As Integer
  x = x + 1
  dble = x * 2
End Function

Function incr(x As Integer, Optional v)
  If IsMissing(v) Then
   incr = x + 1
  Else
   incr = x + v
  End If
End Function

Sub compte()
Dim a As Integer
Static b As Integer
  a = a + 1
  b = b + 1
  MsgBox a & Chr(10) & b
End Sub

Sub tst_proc()
  Dim d  As Integer ' variable locale
  test_proc "abcd", 5
  Call test_proc("def", 3)
  test_proc b:=dble(7), a:="ghi"
  c = 3
  d = dble(c)
  dble (3)        ' cette fonction est appelée comme une procédure
```

```
  dble 3          ' ici aussi
  incr 6, 8       ' incr (6,8) provoque une erreur
  MsgBox d & Chr(10) & c
  MsgBox incr(11) & Chr(10) & incr(17, 3)
  compte
  compte
  compte
  MsgBox (dble(incr(3, 2)))
End Sub
```

VIII. Structure de controle

A. Test

1. Le Si

- **If** <cond> **Then** <instr1>**:** <instr2 >**:** ... **:**<instr n>
- **If** <cond> **Then**
  <instr1>
  <instr2 >
  ...
  <instr n>
  End If
- **If** <cond> **Then**
  <instr alors1>
  ...
  <instr alorsn>
  Else
  <instr sinon1>
  ...
  <instr sinon p>

  End If
- **If** <cond> **Then**
  <instr alors1>
  ...
  <instr alorsn>
  Elseif
  ...
  Elseif
  ...
  Else
  <instr sinon1>
  ...
  <instr sinon p>
  End If

2. Le Cas

Select Case <expression>
  Case <ListeVal1>
  ...

Case <ListeValn>
End Select

Avec <ListeVal> soit

- une valeur ou une expression
- <expr1> **To** <expr2>

B. Boucle Pour

- **For** <cpt> = <dep> **To** <fin>
  <instructions>
  Next <cpt>
- **For** <cpt> = <dep> **To** <fin> **Step** <pas>
  <instructions>
  Next <cpt>
- **For Each** <elt> **In** <ensemble>
  <instructions>
  Next

<cpt> est facultatif dans **Next** <cpt>

C. Boucle Tant que et Jusqu'à

- **Do While** <condition>
  <instructions>
  Loop
- **Do Until** <condition>
  <instructions>
  Loop
- **Do**
  <instructions>
  Loop While <condition>
- **Do**
  <instructions>
  Loop Until <condition>

D. Boucles générales

Do
<instructions>
If <condition> **then Exit Do**
<instructions>
Loop

E. Gestion des Erreurs

- **On Error Goto** <Etiquette>
- **Resume** <Etiquette>
- **Resume Next**

F. Goto, Gosub et autre Exit

Ils existent mais ... Rectangle blanc ...

IX. Méthodes

VBA est orienté objet : les classeurs(WorkBooks ou ActiveWorkBook) , les feuilles (Sheets ou ActiveSheet), les plages (Range) ou les cellules (Cells ou ActiveCell) et Munus ou Barres d'outils ... etc ... sont des objets .

Chaque objet peut exécuter certaines actions définies par des **Méthodes** . Chaque objet possèdent des Propriétés qui contrôlent son comportement ou son apparence.

A. syntaxes

<objet>.<Propriete> = <valeur>

<objet**>.<**Methode> <param. formels>

B. Exemples

Application.Calculation = xlManual pour une propriété.

val = **Application.InputBox(prompt := ‘’**entrez un nombre**’’ , Type :=** 1**)** pour un méthode

X. Boite de dialogue

A. les Boites à messages

<réponse> = **MsgBox (**<mon_messs>,<Bouton>,<Titre>,<fichier d'aide>,<Numérode contexte dans l'aide>**)** Les arguments sont passés dans cet ordre ou remplacés par des espaces ou omis s'ils sont les derniers.

<réponse> = **MsgBox (prompt :=** <mon_messs>,**Buttons :=** <Bouton>,**Title :=** <Titre>, **HelpFile :=** <fichier d'aide>,**Context :=**<Numéro de contexte dans l'aide>**)**

<Bouton> pouvant prendre les valeurs **VbOkOnly,VbOkCancel, VbOkRetryIgnore; VbYesNoCancel, VbYesNo, VbRetryCancel** (Constantes prédéfinies à 0,1,2,3,4,5) éventuellement additionnés aux Logos de signalisation **VbCritical, VbQuestion,VbExclamation, VbInformation** (Constantes prédéfinies à 16,32,48,64)

<réponse> pourra prendre en retour les valeurs **vbOk, VbCancel, VbAbort, VbRetry, VbIgnore, VbYes, VbNo** (Constantes prédéfinies à 1,2,3,4,5,6,7)

B. les Boites à entrées

1. Fonction

<réponse> = **InputBox (prompt :=** <mon_messs>, **Title :=** <Titre>,**Default :=** <Valeur par défaut>, **Xpos :=** <pos Horiz.>, **Ypos :=** <Pos. vert.>, **HelpFile :=** <fichier d'aide>,**Context :=**<Numéro de contexte dans l'aide>**)**

2. Méthode

Set <réponse> = **<**objet**>.InputBox (prompt :=** <mon_messs>, **Title :=** <Titre>,**Default :=** <Valeur par défaut>, **Left :=** <pos Horiz.>, **Top :=** <Pos. vert.>, **HelpFile :=** <fichier d'aide>,**Context :=**<Numéro de contexte dans l'aide>, **Type :=** <type>**)**

<type> est le type de données qui peut être : **0, 1, 2, 4, 8, 16 ou 64** pour une formule, nombre, Chaîne, Booleen, Référence, Erreur, Tableau de Valeur

<objet> est par exemple **Application**

3. Exemple

fichier InputBox.xls

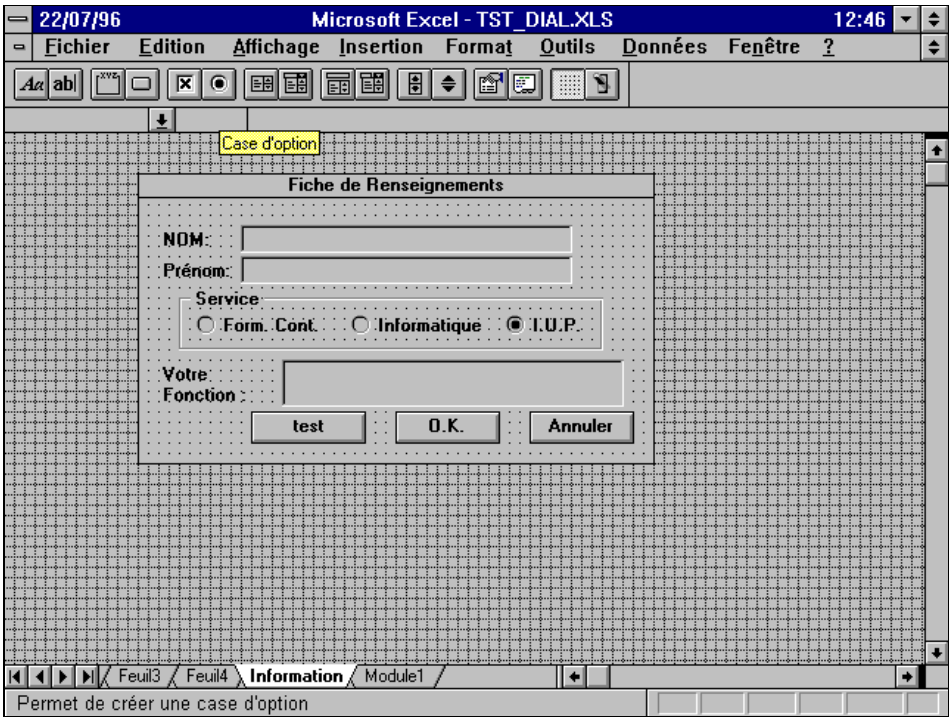
```
Option Explicit
Sub tst_boite()
Dim ch As String
Dim ch2 As Integer
  ch = InputBox("entrez votre nom", "boîte essai", "", 55, 123)
  MsgBox ch
  ch2 = Application.InputBox("entrez un entier", "boîte essai", _
     "", 44, 23, , , 1)
  MsgBox ch2
End Sub
```

C. les Boites personnalisées

fichier **tst_dial.xls**

1. Création d'une feuille de dialogue

du bouton droit sur un onglet inférieur (**feuil<i>**) Puis du bouton gauche sur **Insérer** et sur **Dialog**



Prendre garde de ne faire déborder aucun des 3 « boutons Radio » du cadre Service et renommer cette feuille : Information

2. Texte du Module

```
Option Explicit
Public service As String
Public question As DialogSheet
```

```
Sub mise_en_page()
  Dim rep As Boolean
  Dim chn As String
 ' Sheets("Information").Visible = False
  Set question = DialogSheets("Information")
  question.EditBoxes("nom").Text = ""
  rep = question.Show
  chn = question.EditBoxes("nom").Text & " " & _
        question.EditBoxes("prenom").Text & Chr(10)
  If question.OptionButtons("format").Value = xlOn Then
    chn = chn & "option formation" & Chr(10)
  ElseIf question.OptionButtons("info").Value = xlOn Then
    chn = chn & "option Informatique" & Chr(10)
  ElseIf question.OptionButtons("iup").Value = xlOn Then
    chn = chn & "option IUP" & Chr(10)
  End If
  If rep Then
    chn = chn & "ok"
  Else
    chn = chn & "annuler"
  End If
  MsgBox (chn)
End Sub

Sub tst_QuandClic()
  MsgBox (" test")
End Sub

Sub raz()
  Sheets("Information").Visible = True
End Sub
```

XI. Menus et boutons

fichier **TstMenu.xls**

```
Option Explicit
Public BUtil As Toolbar
Public les_bout As Boolean

Sub retour()
 '
 '
  Sheets("feuil2").Select
  MenuBars(xlWorksheet).Activate
  ActiveWindow.DisplayOutline = True
  If les_bout Then
    Toolbars("jeu_boutons_1_a_145").Delete
    Toolbars("jeu_boutons_212_a_245").Delete
  End If
  BUtil.Delete
'  Toolbars("jeu_bout_1").Delete  'idem a la precedente mais bUtil est global
  MenuBars("util").Delete
  Toolbars(1).Visible = True
  Toolbars(2).Visible = True
  les_bout = False
```

```
End Sub

 Private Sub eff_outils()
Dim result As Integer
  For result = 1 To Toolbars.Count
    Toolbars(result).Visible = False
  Next result
End Sub

 Private Sub mon_menu()
 Dim labarre As MenuBar
 Dim MDonnees As Menu
 Dim mdivers As Menu
  MenuBars.Add Name:="util"
  Set labarre = MenuBars("util")
  labarre.Menus.Add Caption:="&Données"
  Set MDonnees = labarre.Menus("Données")
  labarre.Menus.Add Caption:="D&ivers"
  Set mdivers = labarre.Menus("Divers")
  MDonnees.MenuItems.Add Caption:="essai1", _
                         OnAction:="essai2"  ', _
'                         StatusBar:="essai3"
 mdivers.MenuItems.Add Caption:="retour a la normale   CTRL R", _
                         OnAction:="retour"
 mdivers.MenuItems.Add Caption:="affiche tous les boutons", _
                         OnAction:="tous_les_boutons"
labarre.Activate
End Sub

 Private Sub mes_outils()
  Toolbars.Add Name:="jeu_bout_1"
  Set BUtil = Toolbars("jeu_bout_1")
  BUtil.ToolbarButtons.Add Button:=217, _
                         OnAction:="retour"
    BUtil.ToolbarButtons(1).Name = "retour à l'état initial"
'Toolbars.Add (bUtil)
 With BUtil
    .Visible = True
    .Position = xlTop
  End With
End Sub

Sub tous_les_boutons()
 Dim i As Integer
 Dim Btst As Toolbar
 Dim Btst2 As Toolbar
  If Not les_bout Then
    Toolbars.Add Name:="jeu_boutons_1_a_145"
    Set Btst = Toolbars("jeu_boutons_1_a_145")
    For i = 1 To 145
      Btst.ToolbarButtons.Add Button:=i
    Next i
    With Btst
      .Visible = True
'    .Position = xlTop
      .Left = 20
      .Top = 15
```

```
    .Width = 320
   End With
   Toolbars.Add Name:="jeu_boutons_212_a_245"
   Set Btst2 = Toolbars("jeu_boutons_212_a_245")
   For i = 212 To 245
    Btst2.ToolbarButtons.Add Button:=i
   Next i
   With Btst2
    .Visible = True
'   .Position = xlBottom
    .Left = 347
    .Top = 15
    .Width = 156
   End With
   les_bout = True
  End If
End Sub

Sub menuPerso()
  les_bout = False
  Sheets("feuil1").Select
  ActiveWindow.DisplayOutline = False
  eff_outils
  mon_menu
  mes_outils
End Sub

Sub essai2()
 MsgBox "C'est juste un essai : cliquez sur Divers et Retour"
End Sub
```

XII. VBA dans le tableur

quelques exemples qui peuvent être étudiés en Deug SM1

A. *Premier exemple simple: comportement de suites*

soient les suites

$$u_{n+1} = \frac{u_n^{\,2} - 9}{2u_n - 3} \; et \quad u_{n+1} = \frac{2u_n - 3}{u_n - 2}$$

dont on veut calculer les premiers termes pour différents u_0 .

- Ouvrir un classeur vierge
- sélectionner (par un cliqué et glissé sur leurs numéros) les lignes 4 et 5
- sur **Format** / **Cellule** / **protection** / enlever la sur **verrouillée / Ok**
- sélectionner (par un cliqué et glissé) les cellules A4 à E5
- sur **Format** / **Cellule** / **protection** / mettre la sur **verrouillée / Ok**
- Ecrire « u0 » dans les cellule A4 et « u1 » dans B4
- Ecrire « =svt(B4) » dans la cellule B5 : il y a une erreur NOM? car svt n'est pas encore connue

- sélectionner A5 et B5
- cliquer et glisser le coin inférieur droit de la sélection jusqu'en E5
- sur **Outils** / **protection** / **protéger la feuille**
- sur **Outils / Enregistrer une Macro / Nouvelle Macro** choisir un nom : init
- sélectionner la cellule A5
- sur **Outils** / **protection** / **ôter la protection de la feuille**
- sur **Outils** / **protection** / **protéger la feuille**
- sélectionner A5 et B5
- sur **f**in d'enregistrement
- Vous obtenez une feuille de module contenant:

```
' '
' init Macro
' Macro enregistrée le 25/07/96 par
'
'
Sub init()
    Range("A5").Select
    ActiveSheet.Unprotect
    ActiveSheet.Protect DrawingObjects:=True, Contents:=True, Scenarios _
        :=True
    Range("E4:E5").Select
End Sub
```

- passer dans la feuille module1 qui a été crée automatiquement lors de l'enregistrement de la macro
- Modifiez-la en ajoutant la ligne : **ActiveCell.Value = InputBox("Entrez u0")** ainsi que la fonction **svt()**:

```
Sub init()
  Range("A5").Select
  ActiveSheet.Unprotect
  ActiveCell.Value = InputBox("Entrez u0")
  ActiveSheet.Protect DrawingObjects:=True, Contents:=True, Scenarios _
       :=True
  Range("E4:E5").Select
End Sub

Function svt(x)
  svt = (x * x - 9) / (2 * x - 3)
  'svt = (2 * x - 3) / (x - 2)        ' pour l'autre suite
End Function
```

- Revenir dans la feuil1
- sur **Outils** / **protection** / **ôter la protection de la feuille**
- Sélectionner dans la barre d'outil 1, le bouton dessins (si elle ne s'y trouve pas l'ajouter avec affichage /barre d'outils / standard)
- cliquer dans la nouvelle barre l'icône de bouton

- positionner le bouton dans la grille en effectuant un cliqué et glissé de la cellule milieu de B2 à milieu de C3
- une boite de dialogue propose de donner une liaison avec une macro : init
- changer le nom **Bouton1** en **appuyer ici**
- cliquer sur le nouveau bouton
- donner la valeur de u0
- étendre au besoin la sélection jusqu' a u8 par exemple
- pour visualiser l'autre suite, il suffit de déplacer le commentaire

B. Crible d'Eratostène

Ecriture d'un programme qui inscrit dans la première colonne d'une feuil1 le nombres premiers

on utilisera avantageusement **Cells(**<ligne>**,**<colonne>**)** plutôt que **Range(**<cellule> **)**

C. Compléter la liste des nombres premiers

Se servir de la liste précédente pour rechercher les **nb** nombres premiers suivants en essayant les divisions successives par les nombres premiers

D. Carré Magique

Toujours à partir de la liste précédente construire des carrés magiques selon l'algorithme:

cm(i,j) := (((j + i*l -l-1) mod n+1)+(((i+j*l-l)*n-n-1) mod (n*n))) mod (n*n) + 1

$\forall 0 < i \text{ et } j \leq n \text{ avec } n = prem(l+2) \text{ et } l > 1$

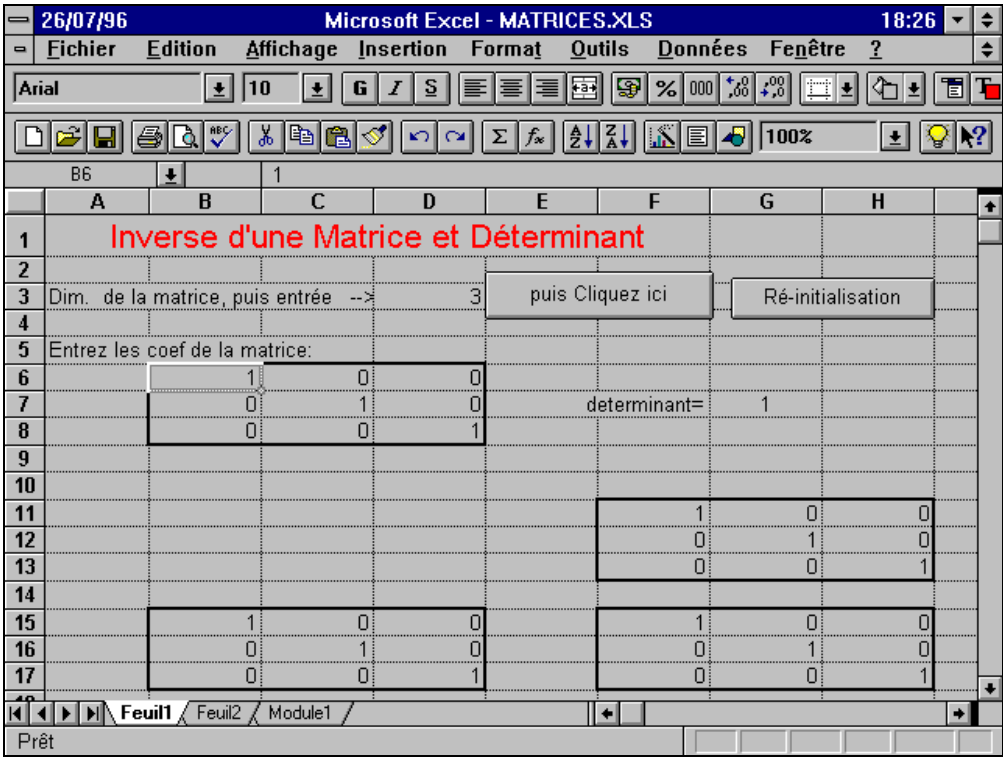
par exemple pour l=2 : (n=7, 2+2 ème nombre premier), le résultat obtenu est :

1	16	31	46	12	27	42
10	25	40	6	21	29	44
19	34	49	8	23	38	4
28	36	2	17	32	47	13
30	45	11	26	41	7	15
39	5	20	35	43	9	24
48	14	22	37	3	18	33

Un module peut être créé pour vérifier qu'il s'agit bien d'un Carré Magique ; ou la vérification peut se faire dans la feuille de calcul..

E. Produit de matrices, Inverse et Déterminant

- Ecrire un module qui en entrée reçois la dimension de la matrice
- prépare la matrice (initialisée à la matrice unité)
- la recopie
- affiche son inverse
- et le produit des 2 pour vérification
- l'utilisateur peut ensuite entrer les coef et obtenir les résultats et la vérification



On pourra enregistrer une Macro et utiliser les formules Excel (dans la feuille de calcul , car dans le module ce ne sont pas les mêmes) :

DETERMAT(<Plage>**)** ; **INDEX(INVERSEMAT(**<Plage>**);**<ligne>**;**<colonne>**)** et
INDEX(PRODUITMAT(<plage matrice 1>**;**<plage matrice 2>**);**<ligne>**;**<colonne>**)**

F. Fichiers

```
Option Explicit

Sub tst_fic()
Dim DonnéesFichier As String

  Open "JOURNAL.TXT" For Append As #1
  Print #1, "Impression d'une ligne dans un fichier."
' Crée un fichier en mode Append s'il n'existe pas déjà.
  Write #1, "Ecriture d'une ligne dans un fichier."
  Close #1
  Open "JOURNAL.TXT" For Input As #1
  Do Until EOF(1)
    Input #1, DonnéesFichier
    MsgBox DonnéesFichier              ' Affiche le texte.
  Loop
  Close #1
End Sub

Sub Enregistre_et_lit_fic()
Dim MesDonnées, NumEnreg As Integer        ' La variable MesDonnées est
                                           ' codée sur 2 octets.
  Open "DONNEES.ENR" For Random As #1
  For NumEnreg = 1 To 10
    MesDonnées = NumEnreg * NumEnreg
    Put #1, NumEnreg, MesDonnées ' Ecrit les données dans le fichier.
```

```
  Next
  MsgBox "les données sont enregistrées" & Chr(10) & " Lecture maintenant"
  For NumEnreg = 1 To 10
    Get #1, NumEnreg, MesDonnées ' Lit les données dans le fichier.
    MsgBox MesDonnées
  Next
  Close #1
End Sub

Sub lecture_fic()
Dim MesDonnées, NumEnreg As Integer

  Open "DONNEES.ENR" For Random As #1
  For NumEnreg = 1 To 10
    Get #1, NumEnreg, MesDonnées ' Lit les données dans le fichier.
    MsgBox MesDonnées
  Next
  Close #1
End Sub
```

G. Triangle de Pascal

H. Biorythmes

I. Tracé de courbes y=f(x)

J. Tables de Pythagore.

K. Traducteur de Modules

Un module écrit en VBA peut être sauvegardé sous forme de texte grâce à **Fichier / Enregistrer sous / Type de fichier : Code Basic (texte)**

Un programme (module) peut lire ce fichier texte (un module aussi) mot par mot et, à l'aide de ListeVba.xls (qui contient la liste des traductions Français/Anglais des Mots clés de VBA) fournir un module « traduit » soit en anglais soit en français (selon le source).

L. Divers

Les fichiers XL.exe , mathexc4.exe et mathexc5.exe sont des exécutables Dos qui décompactent diverses applications excel.

XIII. Annexes

Tirées de **vba_xl.hlp**

A. Objet : Range

Représente une cellule, une ligne, une colonne, une sélection de cellules contenant un ou plusieurs blocs de cellules contigus ou bien une plage 3D.

Membres d'accès

Voici plusieurs exemples d'utilisation des principales méthodes et propriétés d'accès de l'objet Range.

1. Méthode Range

Utilisez la méthode Range pour renvoyer une cellule ou une plage de cellules. L'exemple suivant place la valeur de la cellule A1 dans la cellule A5.

```
Worksheets("Feuil1").Range("A5").Value =   Worksheets("Feuil1").Range("A1").Value
```

L'exemple suivant remplit la plage A1:H8 avec des nombres aléatoires en définissant la formule de chacune des cellules de cette plage. Lorsqu'elle est utilisée sans qualificateur d'objet (objet précédant le point), la méthode Range renvoie une plage de la feuille active. Si celle-ci n'est pas une feuille de calcul, la méthode échoue. Utilisez la méthode Activate pour activer une feuille de calcul avant d'employer la méthode Range sans qualificateur d'objet explicite.

```
Worksheets("Feuil1").Activate
Range("A1:H8").Formula = "=RAND()"      'La plage est dans la feuille active
```

L'exemple suivant efface le contenu de la plage nommée «Critère».

```
Worksheets(1).Range("Critère").ClearContents
```

Si vous utilisez un argument de type texte pour l'adresse de la plage, vous devez spécifier l'adresse en style de référence A1 (vous ne pouvez pas employer le style de référence L1C1).

2. Méthode Cells

Utilisez la méthode Cells pour renvoyer une cellule en spécifiant la ligne et la colonne. L'exemple suivant affecte la valeur 24 à la cellule A1.

```
Worksheets(1).Cells(1, 1).Value = 24
```

L'exemple suivant définit la formule de la cellule A2.

```
ActiveSheet.Cells(2, 1).Formula = "=SUM(B1:B5)"
```

Bien que vous puissiez renvoyer la cellule A1 à l'aide de Range("A1"), la méthode Cells peut se révéler plus pratique, car elle vous permet d'utiliser une variable pour la ligne ou la colonne. L'exemple suivant crée des titres de colonne et de ligne dans la feuille de calcul nommée «Feuil1». Remarquez qu'une fois la feuille de calcul activée, la méthode Cells peut être utilisée sans déclaration de feuille explicite (elle renvoie une cellule de la feuille active).

```
Sub InitialiserTable()
Worksheets("Feuil1").Activate
For année = 1 To 5
        Cells(1, année + 1).Value = 1990 + année
Next année
'
For leTrimestre = 1 To 4
        Cells(leTrimestre + 1, 1).Value = "T" & leTrimestre
Next leTrimestre
End Sub
```

Bien que vous puissiez modifier les références de style A1 à l'aide de fonctions Visual Basic de type chaîne, il est beaucoup plus facile, et préférable en termes de programmation, d'employer la notation Cells(1, 1).

3. Méthodes Cells et Range

Vous pouvez également utiliser la méthode Cells pour renvoyer une partie d'une plage en spécifiant la ligne et la colonne par rapport au coin supérieur gauche de la plage. L'exemple suivant définit la formule de la cellule C5.

```
Worksheets(1).Range("C5:C10").Cells(1,1).Formula = "=RAND()"
```

Vous pouvez aussi utiliser la méthode Cells à l'intérieur de la méthode Range pour spécifier les cellules de début et de fin de la plage. L'exemple suivant définit le style de trait de la bordure des cellules A1 à J10.

```
With Worksheets(1)
   .Range(.Cells(1, 1), .Cells(10, 10)).Borders.LineStyle = xlThick
End With
```

Remarquez la présence d'un point devant chacune des occurrences de la méthode Cells. Ce point est indispensable pour que le résultat de l'instruction With qui précède soit appliqué à la méthode Cells (dans le cas présent, pour indiquer que les cellules appartiennent à la feuille de calcul 1). En l'absence de ce point, la méthode Cells renverrait des cellules de la feuille active.

4. Méthode Offset

La méthode Offset renvoie une plage qui est décalée par rapport à une autre plage. L'exemple suivant sélectionne la cellule située trois lignes en dessous et une colonne à droite de la cellule supérieure gauche de la sélection courante. Dans la mesure où vous ne pouvez pas sélectionner de cellule hors de la feuille active, vous devez d'abord activer la feuille de calcul.

```
Worksheets("Feuil1").Activate  'Sélection impossible à moins que la 'feuille soit active
Selection.Offset(3, 1).Range("A1").Select
```

5. Méthode Union

Utilisez la méthode Union et la méthode Range pour renvoyer des plages constituées d'au moins deux blocs de cellules contigus. L'exemple suivant crée un objet appelé maPlageMultiZone, le définit en tant qu'union des plages A1:B2 et C3:D4, puis le sélectionne.

```
Dim r1, r2, maPlageMultiZone As Range
Worksheets("Feuil1").Activate
Set r1 = Range("A1:B2")
Set r2 = Range("C3:D4")
Set maPlageMultiZone = Union(r1, r2)
maPlageMultiZone.Select
```

6. Méthode Areas

La méthode Areas est très utile si vous travaillez avec des sélections qui contiennent plusieurs zones. Cette méthode divise une telle sélection en différents objets Range qu'elle renvoie ensuite sous la forme d'une collection. Vous pouvez appliquer la propriété Count à la collection ainsi renvoyée, de manière à rechercher une sélection contenant plusieurs zones, comme dans l'exemple suivant:

```
Sub PasDeSelectionMultiZone()
        nbreZonesSelectionnees = Selection.Areas.Count
        If nbreZonesSelectionnees > 1 Then
                MsgBox "Vous ne pouvez pas exécuter cette commande " & _
                        "sur des sélections contenant plusieurs zones"
        End If
End Sub
```

B. MenuBar, objet

Représente une barre de menus prédéfinie ou personnalisée. Pour plus d'informations sur la structure des menus, reportez-vous à la rubrique d'aide Structure des menus.

Membres d'accès

L'objet MenuBar est un membre de la collection MenuBars, qui contient l'ensemble des barres de menus disponibles dans Microsoft Excel. Utilisez la méthode Add pour créer une barre de menus et l'ajouter à la collection. Utilisez la méthode Activate pour afficher une barre de menus. Pour accéder à un membre donné de la collection, utilisez la méthode MenuBars en indiquant comme argument le numéro d'index ou l'intitulé de la barre de menus voulue.

La liste suivante présente les constantes prédéfinies qui peuvent être utilisées comme numéro d'index de la barre de menus.

Constante	Description
xlWorksheet	Feuille de calcul, feuille macro et feuille boîte de dialogue
xlChart	Graphique
xlModule	Module Visual Basic
xlNoDocuments	Aucun document n'est ouvert
xlInfo	Fenêtre Info
xlWorksheetShort	Menu Feuille de calcul abrégé (pour compatibilité avec Microsoft Excel version 3)
xlChartShort	Menu Graphique abrégé (pour compatibilité avec Microsoft Excel version 3)
xlWorksheet4	Ancienne barre de menus de feuille de calcul (pour compatibilité avec Excel 4)
xlChart4	Ancienne barre de menus de graphique (pour compatibilité avec Microsoft Excel version 4)

Microsoft Excel affiche automatiquement la barre de menus prédéfinie appropriée à la feuille active. Microsoft Excel affiche la barre de menus en fonction du type de la feuille active. Si vous créez une barre de menus personnalisée et si vous l'affichez à l'aide du code Visual Basic, Microsoft Excel cesse d'afficher une barre de menus adaptée à la feuille active et c'est à votre code qu'il revient d'activer et de désactiver (au besoin) la barre de menus à mesure que le type de la feuille active change.

L'exemple suivant ajoute une nouvelle commande au bas du menu Fichier de la barre de menus du module Visual Basic.

```
MenuBars(xlModule).Menus("Fichier").MenuItems.Add "&Chercher"
```

L'exemple suivant crée, dans la feuille de calcul 1, une table qui contient les intitulés de tous les menus de l'ensemble des barres de menus de l'application. Les titres des colonnes sont constitués par les intitulés des barres de menus, tandis que les menus qui composent ces différentes barres apparaissent dans les cellules des colonnes correspondantes.

```
Sub EnumererBarresDeMenus()
    Worksheets(1).Activate
    c = 1
    For Each mb In MenuBars
        Cells(1, c) = mb.Caption
        i = 2
        For Each mn In mb.Menus
            Cells(i, c) = mn.Caption
            i = i + 1
        Next
        c = c + 1
    Next
End Sub
```

C. Add, méthode (collection MenuItems)

Ajoute un nouvel élément de menu au menu spécifié. Permet également de rétablir un élément de menu prédéfini qui a été supprimé dans Microsoft Excel. Renvoie un objet MenuItem. Utilisez la méthode AddMenu pour ajouter un intitulé de menu secondaire.

Syntaxe

objet.Add(caption, onAction, shortcutKey, before, restore, statusBar, helpContextID, helpFile)

objet

Requis. L'objet MenuItems.

caption

Requis. Le texte de la commande (définit la valeur initiale de la propriété Caption pour la nouvelle commande). Insérez un signe & avant la lettre que vous souhaitez souligner. Pour créer une barre séparatrice, utilisez un tiret unique ("-") comme intitulé.

onAction

Facultatif. Le nom de la macro qui est exécutée lorsqu'un nouveau élément de menu est sélectionné.

shortcutKey

Facultatif. N'est utilisé que pour Macintosh. Spécifie en tant que texte la touche de raccourci pour l'élément du menu.

before

Facultatif. Spécifie l'élément de menu avant lequel l'élément de menu est ajouté. Il peut s'agir d'un nombre (1 pour l'insérer au début du menu, par exemple) ou un intitulé d'un autre élément de menu existant (dans le langage de la macro, sans signe &), ou encore une référence à l'élément de menu.

restore

Facultatif. Si True, Microsoft Excel rétablit l'élément de menu intégré précédemment supprimé et nommé caption
. Si False ou omis, Microsoft Excel ajoute un nouvel élément de menu. L'élément de menu rétabli est placé à la fin du menu, à moins que vous n'utilisiez l'argument before pour définir son emplacement.

statusBar

Facultatif. Spécifie le texte à afficher dans la barre d'état lorsque l'utilisateur parcoure ou sélectionne l'élément de menu. S'il est omis, c'est le texte de barre d'état affecté à la macro qui est utilisé.

helpContextID

Facultatif. Spécifie l'identificateur contextuel pour la rubrique d'aide associée à l'élément de menu. S'il est omis, c'est l'identificateur contextuel affecté à la macro qui est utilisé.

helpFile

Facultatif. Spécifie le nom du fichier d'Aide qui contient la rubrique d'Aide de l'élément de menu. S'il est omis, c'est le nom du fichier d'Aide affecté à la macro qui est utilisé.

Remarques

La définition des arguments statusBar, helpContextID et helpFile remplace celle des options de la macro spécifiée par l'argument onAction.
Cependant, si les options de la macro sont modifiées après l'ajout de l'élément de menu, leurs arguments d'aide personnalisée deviennent alors prioritaires.
Cet exemple ajoute une nouvelle commande au début du menu ? (Aide) de chaque barre de menus.

```
For Each mb in MenuBars
        mb.Menus("Aide").MenuItems.Add _
                caption:="Lisez-moi", _
                onAction:="Lisez_moi", _
                before:=1
Next mb
```

D. Delete, méthode

Supprime l'objet. Syntaxe 2 s'applique seulement aux objets Range.

Syntaxe 1

objet.Delete

Syntaxe 2

objet.Delete(shift)

objet

Requis. L'objet auquel s'applique cette méthode. Syntaxe 2 s'applique seulement aux objets Range.

shift

Facultatif. Spécifie comment décaler les cellules de façon à remplacer celles supprimées (soit xlToLeft ou xlUp). Si cet argument est omis, Microsoft Excel sélectionne une valeur par défaut basée sur la forme de la plage.

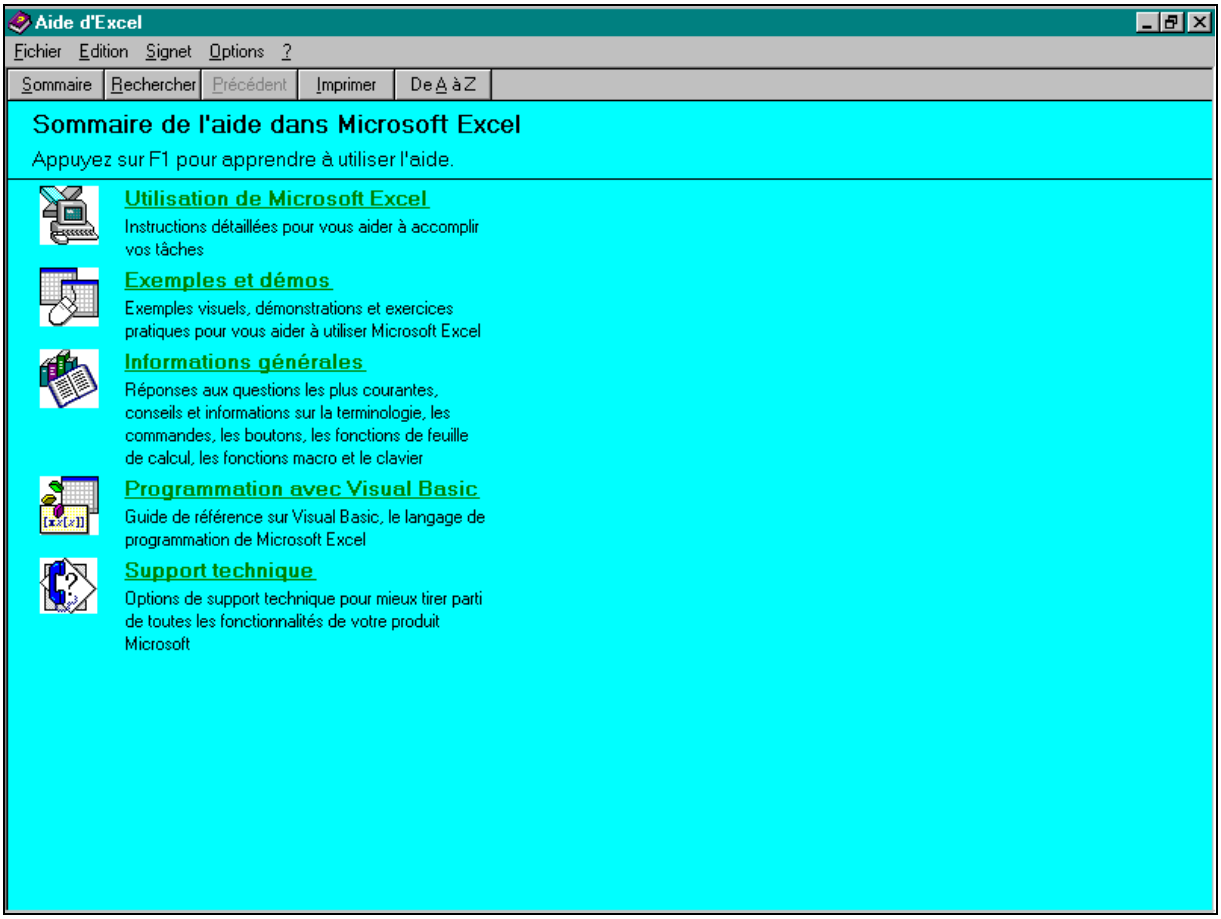
Remarques

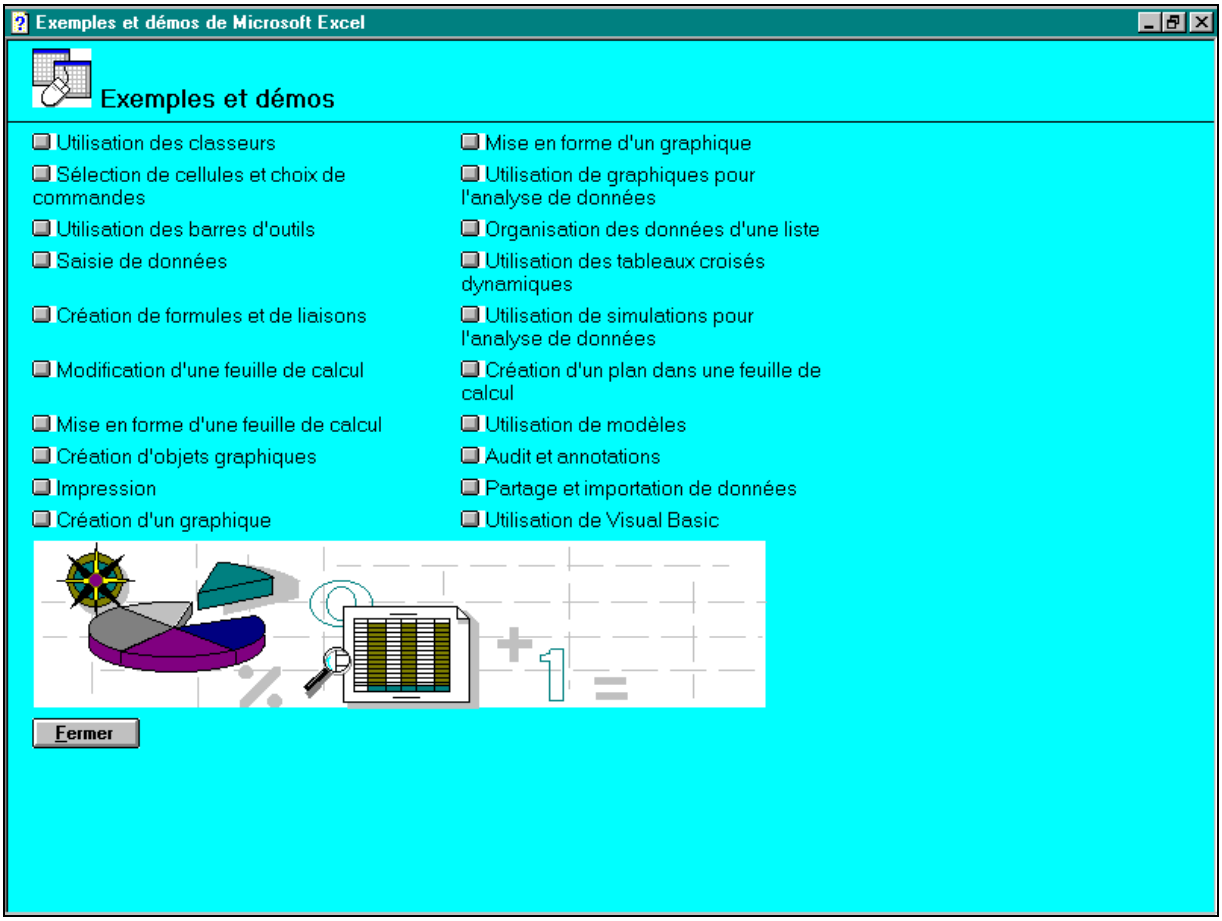
Les tentatives pour supprimer un objet Toolbar ou MenuBar prédéfini échouent mais n'engendrent pas d'erreur. Cela vous permet d'utiliser une boucle For Each pour supprimer toutes les barres d'outils et de menus personnalisées.
Supprimer un objet Point ou LegendKey supprime la série dans son intégralité.
Vous pouvez supprimer les propriétés d'un document personnalisé, mais pas celles d'un document intégré.

E. Autoformation

Des aides et démonstrations sont fournies avec le pack Office :

le fichier mainxl.hlp propose :


Aide d'Excel
Fichier Edition Signet Options ?
Sommaire Rechercher Précédent Imprimer De A à Z
Sommaire de l'aide dans Microsoft Excel
Appuyez sur F1 pour apprendre à utiliser l'aide.
Utilisation de Microsoft Excel
Instructions détaillées pour vous aider à accomplir vos tâches
Exemples et démos
Exemples visuels, démonstrations et exercices pratiques pour vous aider à utiliser Microsoft Excel
Informations générales
Réponses aux questions les plus courantes, conseils et informations sur la terminologie, les commandes, les boutons, les fonctions de feuille de calcul, les fonctions macro et le clavier
Programmation avec Visual Basic
Guide de référence sur Visual Basic, le langage de programmation de Microsoft Excel
Support technique
Options de support technique pour mieux tirer parti de toutes les fonctionnalités de votre produit Microsoft



XIV. Débogage

L'exécution du module :

```
Option  Explicit

Sub test_debug()
Dim i As Integer
For i = 11 To 20    ' La boucle est effectuée 10
                    ' fois.
   ' Imprime chaque valeur sur une nouvelle ligne.
   Debug.Print i

Next i

For i = 11 To 20    ' La boucle est effectuée 10
                    ' fois.
   ' Imprime les valeurs les unes à côté des autres sur la même ligne.

Debug.Print i;
Next i

' Imprime 10 espaces avant.
Debug.Print Spc(10); "Bonjour tout le monde"
' Imprime dans la colonne 20.
Debug.Print Tab(20); "Ceci est un test"
```

```
' Imprime 2 zones d'impression plus loin.
Debug.Print "Bonjour"; Tab; Tab; "tout le monde"
End Sub
Sub test_debug()
Dim i As Integer
For i = 11 To 20    ' La boucle est effectuée 10
                    ' fois.
   ' Imprime chaque valeur sur une nouvelle ligne.
   Debug.Print i

Next i

For i = 11 To 20    ' La boucle est effectuée 10
                    ' fois.
   ' Imprime les valeurs les unes à côté des autres sur la même ligne.

Debug.Print i;
Next i

' Imprime 10 espaces avant.
Debug.Print Spc(10); "Bonjour tout le monde"
' Imprime dans la colonne 20.
Debug.Print Tab(20); "Ceci est un test"

' Imprime 2 zones d'impression plus loin.
Debug.Print "Bonjour"; Tab; Tab; "tout le monde"
End Sub
```

peut être obtenue dans une fenêtre spécifique, accessible grâce à **affichage / fenêtre Débogage** (<Ctrl>+G).

XV. Programme en Deug SM1

La réflexion est ouverte :

Quels seront les choix parmi les éléments présentés ci-avant ?

envoyez-moi vos idées par E-Mail à mailliet@lifl.fr