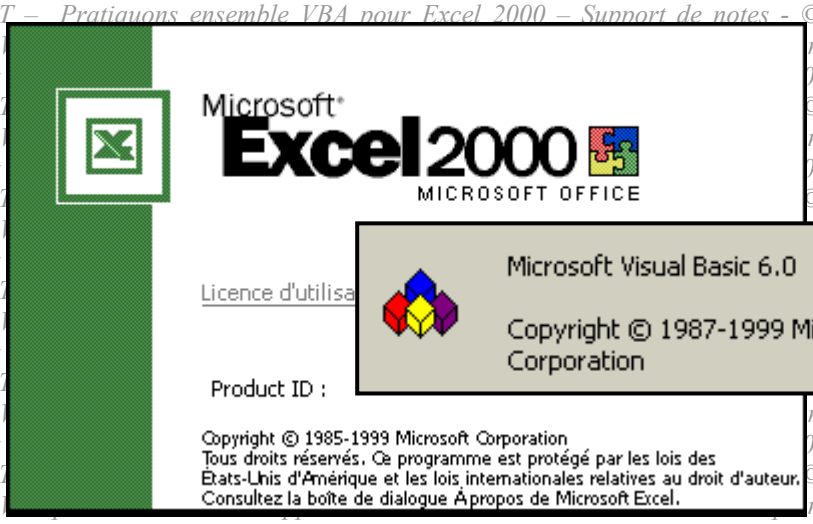
PRATIQUONS ENSEMBLE



SUPPORT DE NOTES

Nom : .....

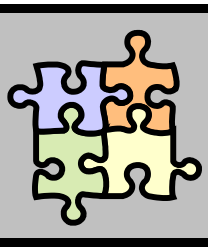
Prénom : .....

Attention !

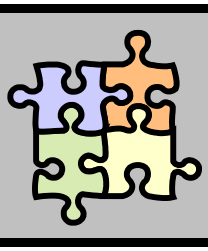
Cet outil est un support de notes.

Il ne remplace, en aucun cas, un outil de formation
ou la documentation fournie avec le logiciel.

TABLE DES MATIÈRES



INTRODUCTION



Excel 2000 est un outil de développement fournissant simultanément les avantages des feuilles de calcul et d'un outil de programmation : le langage Visual Basic Edition Application ou VBA.

Les « programmes » réalisés libèrent l'utilisateur de l'exécution de tâches répétitives.

Il existe deux types de « programmes » :
- Les **fonctions personnalisées**
- Les **macros** ou **procédures**

1. LES FONCTIONS PERSONNALISÉES

Une fonction personnalisée est un programme, écrit en VBA, qui effectue un ou plusieurs calculs sur des données et renvoie un résultat.
Une fonction est généralement utilisée dans une cellule d'une feuille de calcul (de la même façon que vous utilisez la fonction Somme).
Elle peut aussi être utilisée dans une procédure.

2. LES MACROS OU PROCÉDURES

Une macro ou procédure est un programme, écrit en VBA, mémorisant une suite d'actions (ou d'instructions) réalisées sur un classeur, sur une feuille ou encore sur une cellule ou un groupe de cellules.
Contrairement aux fonctions, les macros ou procédures ne retournent aucune valeur lorsqu'on les exécutent. Elles réalisent cependant les différentes actions mémorisées (mise en forme, saisie, modification, suppression, déplacement…).

Une macro peut être entièrement réalisée par enregistrement. Pour cela, il faut :
- lancer l'enregistreur de macros,
- réaliser, dans l'ordre, les différentes actions souhaitées, en utilisant les commandes des menus, les boutons des barres d'outils et les touches du clavier,
- arrêter l'enregistreur de macros.

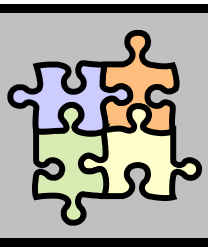
Une macro peut être entièrement écrite par le concepteur. Pour cela, il faut utiliser l'éditeur VBA comme environnement de développement.

3. L'ÉDITEUR VBA ET LES MODULES

L'éditeur VBA permet de visualiser les fonctions personnalisées ou les macros créées. Celles-ci se trouvent dans des **modules** (qui se présentent comme des documents d'un traitement de texte).

Page volontairement vide

LES FONCTIONS PERSONNALISÉES



1. CRÉER UNE FONCTION PERSONNALISÉE

1. 1. Généralités

La syntaxe d'une fonction est :

```
Function nom_fonction(argument1,argument2…)
        opération(s) à exécuter
End Function
```

- Créez le tableau ci-contre (mettre les cellules **B2** à **C7** en format Nombre, 2 décimales et séparateur de milliers).
- Enregistrez le classeur sous le nom **Ventes**.

	A	B	C
1		Tarbes	Pau
2	Ventes	100000	93000
3	Coûts production	23000	21000
4	Coûts distribution	5300	4929
5	Marge brute		
6	Frais généraux		
7	Marge nette		
8			

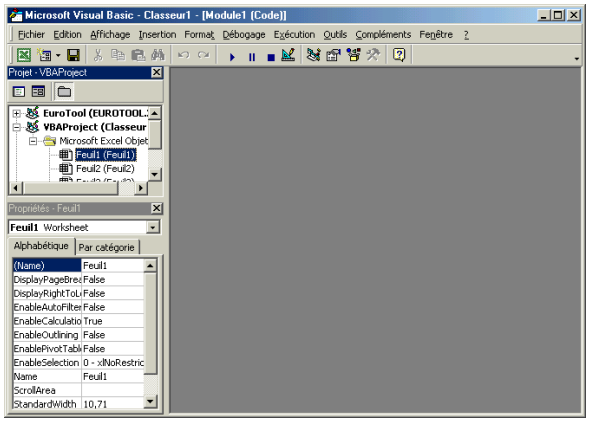
1. 2. Création d'une fonction simple

Nous allons créer une fonction permettant de calculer la Marge brute à partir des Ventes, du Coût de fabrication et du Coût de distribution : mbrute = ventes - (production + distribution)

- **Outils**
- **Macro**
- **Visual Basic Editor**

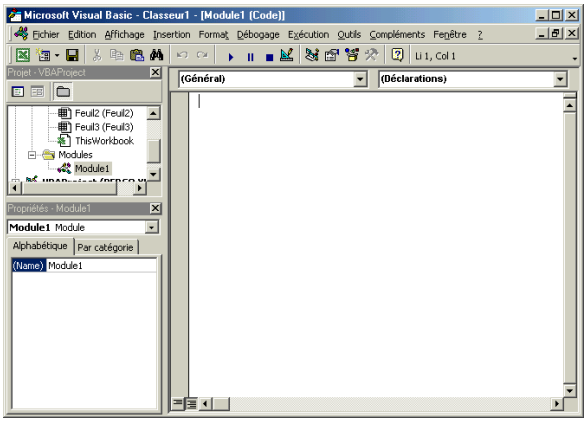
La fenêtre **Microsoft Visual Basic** apparaît :

Remarque : cette fenêtre est accessible par le raccourci clavier **Alt** + **F11**



- **Insertion**
- **Module**

Une fenêtre **Module1** apparaît sur le côté droit.

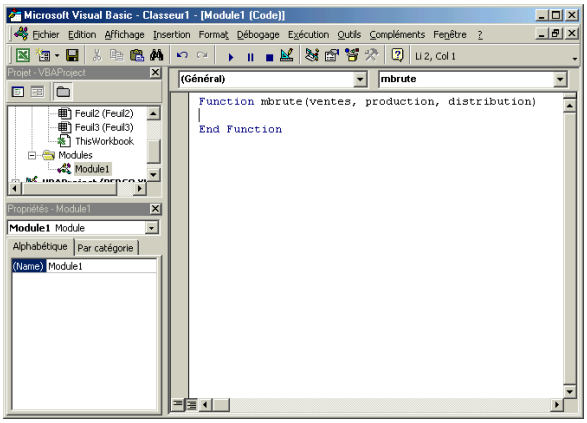


- Dans la fenêtre du module, saisissez le nom de la fonction et ses arguments :

Function mbrute(ventes,production,distribution)

- Appuyez sur la touche Entrée

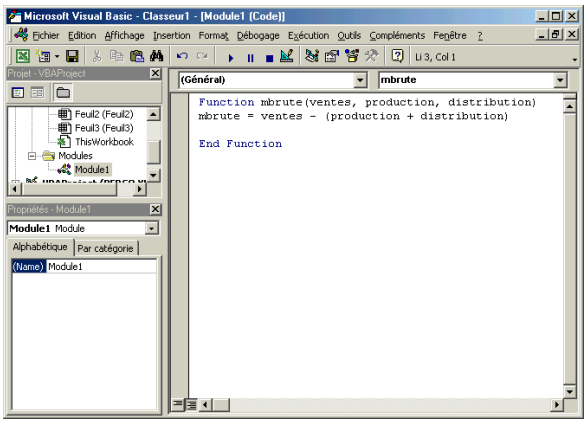
Les mots **End Function** apparaissent automatiquement.



- Saisissez l'opération à exécuter :

mbrute=ventes-(production+distribution)

- Appuyez sur la touche Entrée

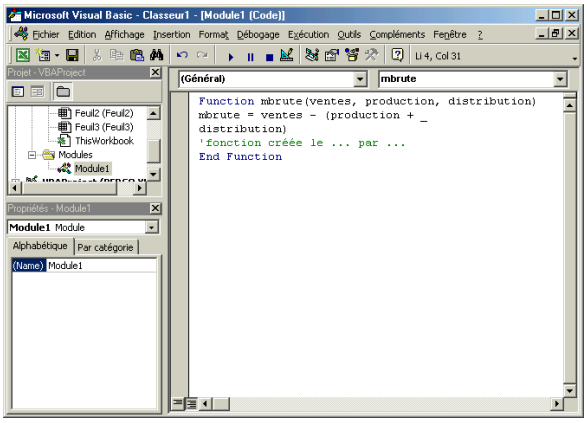


Remarques :

Si le code est trop long, un retour à la ligne est possible. Pour cela, saisissez un espace, le symbole _ et Entrée. Vous pouvez saisir la suite du code sur la nouvelle ligne.

Un commentaire peut être saisi en le faisant précéder de l'apostrophe '.

Il est recommandé de tabuler devant chaque ligne de code pour faciliter la lecture.

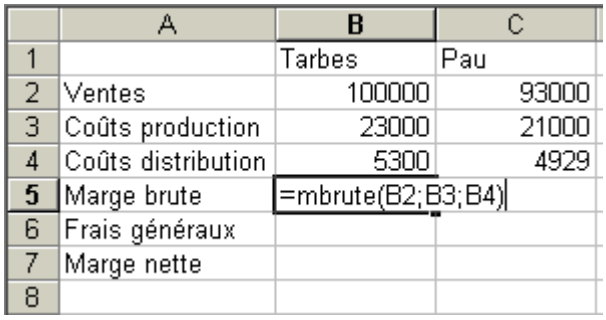


1. 3. Utilisation de la fonction

- Fermez la fenêtre **Microsoft Visual Basic**

Le tableau Excel réapparaît.

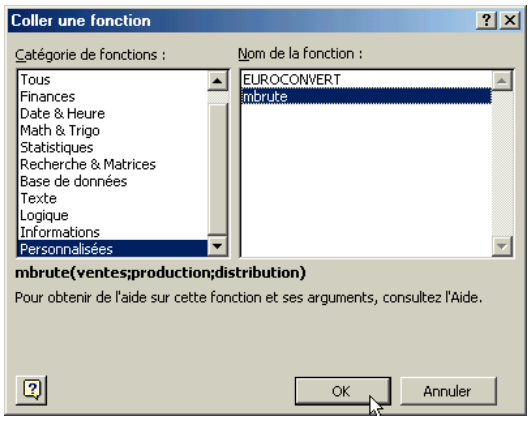
- Dans la cellule **B5**, saisissez **=mbrute(**
- cliquez sur la cellule **B2** des ventes et tapez ;
- cliquez sur la cellule **B3** des coûts de fabrication et tapez ;
- cliquez sur la cellule **B4** des coûts de distribution et tapez)



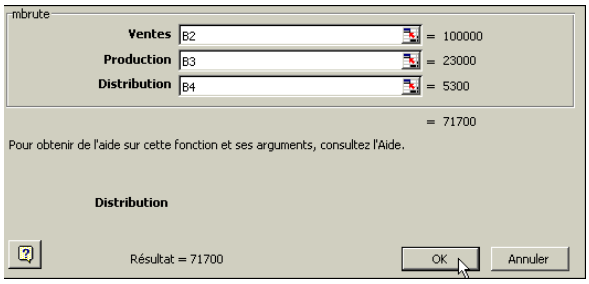
	A	B	C
1		Tarbes	Pau
2	Ventes	100000	93000
3	Coûts production	23000	21000
4	Coûts distribution	5300	4929
5	Marge brute	=mbrute(B2;B3;B4)	
6	Frais généraux		
7	Marge nette		
8			

- Validez avec Entrée.
- Recopiez la formule dans la cellule voisine.
- Enregistrez le classeur sous le nom **Fonctions**.

Remarque :
Il est possible d'accéder à la fonction personnalisée (depuis une cellule vide) en utilisant l'**Assistant Fonction**  .
Celle-ci se trouve dans la catégorie **Personnalisées**.



L'Assistant Fonction s'utilise de la même façon qu'avec une fonction intégrée (comme la fonction Somme, par exemple).



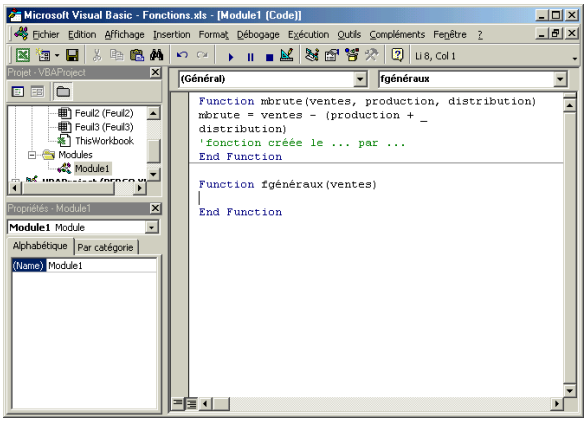
1. 4. Création d'une fonction conditionnelle à une condition

Nous allons créer une fonction permettant de déterminer les Frais généraux en fonction des Ventes : frais généraux = ventes * taux.
Le taux est de 10 % lorsque les ventes sont supérieures à 100 000 F
Sinon, le taux est de 5 %

Le code VBA est :

```
If Condition Then
        Action à réaliser si Condition vérifiée
Else
        Action à réaliser si Condition non vérifiée
End If
```

- **Outils**
- **Macro**
- **Visual Basic Editor**

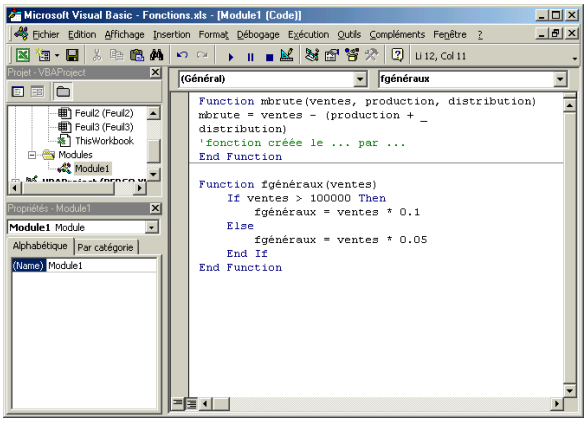


- A la fin de la fenêtre du **Module1**, saisissez la fonction :

Function fgénéraux(ventes)

- Appuyez sur la touche Entrée

Les mots **End Function** apparaissent automatiquement.



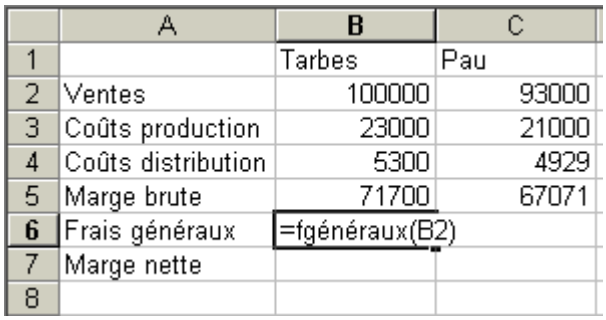
- Saisissez l'opération à exécuter :

If ventes > 100000 **Then** *(Entrée)*
fgénéraux=ventes*0.1 *(Entrée)*
Else *(Entrée)*
fgénéraux=ventes*0.05 *(Entrée)*
End If *(Entrée)*

- Fermez la fenêtre **Microsoft Visual Basic**

1. 5. Utilisation de la fonction

- Dans la cellule **B6**, saisissez **=fgénéraux(**
- Cliquez sur la cellule **B2** des ventes et tapez **)**



	A	B	C
1		Tarbes	Pau
2	Ventes	100000	93000
3	Coûts production	23000	21000
4	Coûts distribution	5300	4929
5	Marge brute	71700	67071
6	Frais généraux	=fgénéraux(B2)	
7	Marge nette		
8			

- Validez avec Entrée.
- Recopiez la formule dans la cellule voisine.
- Enregistrez le classeur.

1. 6. Exercice

Sur les mêmes principes, créez la fonction permettant de calculer la Marge nette (différence entre la Marge brute et les Frais généraux).
Testez cette fonction dans les cellules **B7** et **B8**.

1. 7. Création d'une fonction conditionnelle à plusieurs conditions

Nous allons créer une fonction permettant de déterminer les Frais généraux en fonction des Ventes.
Le taux est de 5 % lorsque les ventes sont inférieures à 80 000 F
Le taux est de 7 % lorsque les ventes sont comprises entre 80 000 F et 100 000 F
Le taux est de 10 % lorsque les ventes sont supérieures à 100 000 F

Le code VBA est :

```
If Condition1 Then
        Action à réaliser si Condition1 vérifiée
ElseIf Condition2 Then
        Action à réaliser si Condition2 vérifiée
ElseIf Condition3 Then
        Action à réaliser si Condition3 vérifiée
Else
        Action à réaliser si aucune condition n'est vérifiée
End If
```

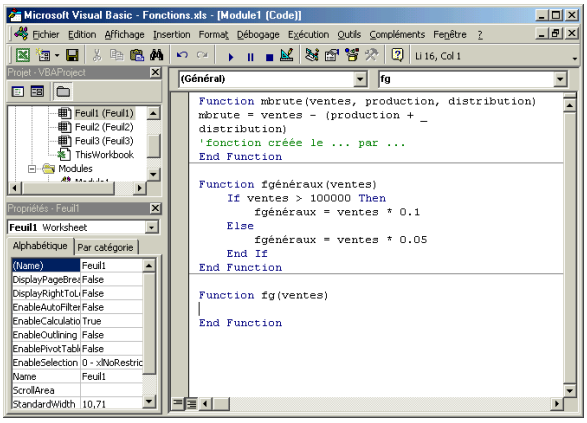
- **Outils**
- **Macro**
- **Visual Basic Editor**

- A la fin de la fenêtre du **Module1**, saisissez la fonction :

Function fg(ventes)

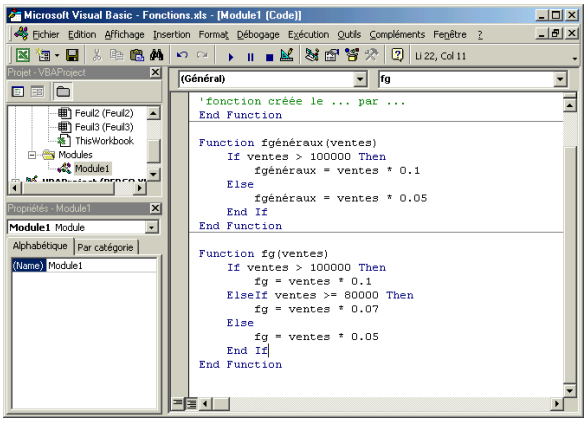
- Appuyez sur la touche Entrée

Les mots **End Function** apparaissent automatiquement.



- Saisissez l'opération à exécuter :

If ventes > 100000 **Then** *(Entrée)*
fg=ventes*0.1 *(Entrée)*
ElseIf Ventes >= 80000 **Then** *(Entrée)*
fg=ventes*0.07 *(Entrée)*
Else *(Entrée)*
fg=ventes*0.05 *(Entrée)*
End If *(Entrée)*



- Fermez la fenêtre **Microsoft Visual Basic**

1. 8. Utilisation de la fonction

- Dans la cellule **B6**, saisissez **=fg(**
- Cliquez sur la cellule **B2** des ventes et tapez **)**

	A	B	C
1		Tarbes	Pau
2	Ventes	100000	93000
3	Coûts production	23000	21000
4	Coûts distribution	5300	4929
5	Marge brute	71700	67071
6	Frais généraux	=fg(B2)	
7	Marge nette		
8			

- Validez avec Entrée.
- Recopiez la formule dans la cellule voisine.
- Enregistrez et fermez le classeur.

Remarque :

Pour réaliser cette fonction, nous aurions pu utiliser le code VBA ci-contre :

Pour plus d'informations sur **Select Case**, voir page 34)

```
Function fg(ventes)
        Select Case ventes
                Case Is > 100000
                        fg=ventes*0.1
                Case Is >= 80000
                        fg=ventes*0.07
                Case Else
                        fg=ventes*0.05
        End Select
End Function
```

1. 9. Exercice

A partir de la fenêtre **Microsoft Visual Basic**, consultez l'aide et les exemples des fonctions :

- Iif
- Choose
- Switch

Remarque : vous pouvez imprimer l'aide de chacune des fonctions pour compléter votre support de notes.

2. UTILISER UNE FONCTION PERSONNALISÉE DANS UN NOUVEAU CLASSEUR

Une question se pose : Peut-on utiliser une fonction personnalisée dans un nouveau classeur ? Nous allons chercher la réponse !

- Créez un nouveau classeur.
- Saisissez l'exemple ci-contre.

	A	B
1		Lannemezan
2	Ventes	55000
3	Frais généraux	
4		

- Dans la cellule **B3**, saisissez **=fg(**
- cliquez sur la cellule **B2** des ventes et tapez)
- Validez avec Entrée.

	A	B
1		Lannemezan
2	Ventes	55000
3	Frais généraux	#NOM?
4		
5		

Le résultat du calcul est #NOM? : la fonction n'est pas connue d'Excel dans ce classeur.

Nous pouvons donc en conclure qu'une fonction personnalisée n'est utilisable que dans le classeur dans lequel elle a été créée.

Il est néanmoins possible de généraliser une fonction personnalisée à l'ensemble des classeurs.

- Fermez ce classeur.

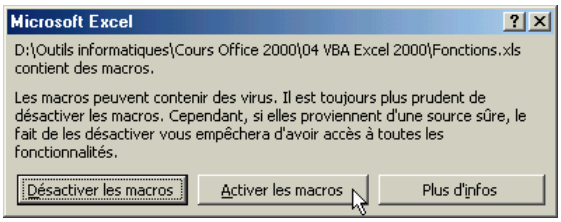
3. GÉNÉRALISER UNE FONCTION PERSONNALISÉE

Pour généraliser une fonction personnalisée (et l'utiliser de n'importe quel classeur), il faut :
- mémoriser cette fonction comme macro complémentaire,
- activer la macro complémentaire.

3. 1. Ouverture du classeur contenant la fonction

- Ouvrez le classeur **Fonctions.xls** contenant les fonctions personnalisées que vous venez de créer.

Lors de l'ouverture de ce classeur, Excel affiche un message vous alertant que ce fichier contient des macros (dans notre cas, ce fichier contient des fonctions, Excel ne fait pas la différence).



Attention : Tout « programme » peut contenir un virus !

Pas de danger avec le contenu de ce classeur puisque vous en êtes le créateur.

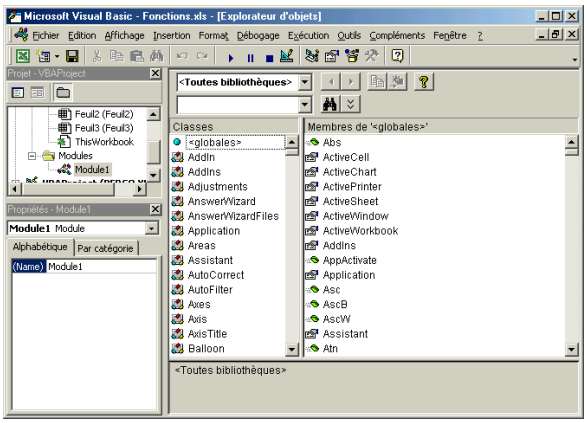
- Vous pouvez cliquer sur le bouton **Activer les macros**.

Remarque : il est recommandé de ne pas désactiver la case « Toujours demander confirmation avant d'ouvrir des classeurs contenant de macros », surtout si vous manipulez des fichiers Excel que vous n'avez pas créé.

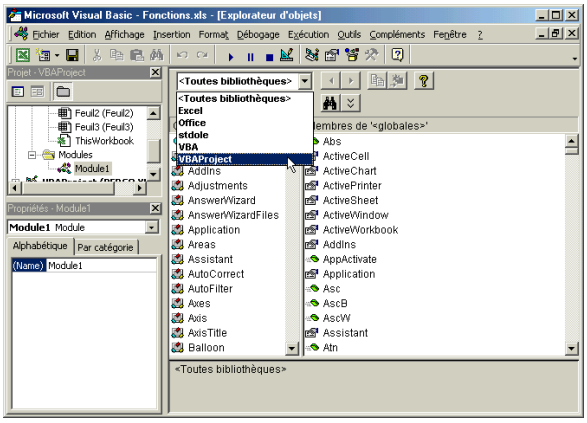
3. 2. Ajout de commentaires

Lorsque nous avons utilisé la fonction depuis l'Assistant Fonction, celui-ci n'affiche aucun commentaire (voir page 9). L'utilisateur ne sait pas forcément ce que fait la fonction.

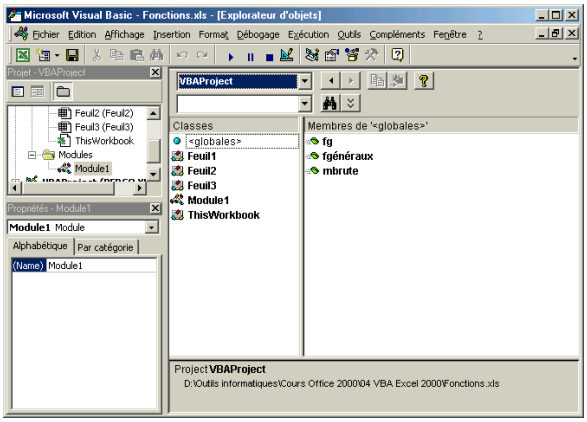
- Ouvrez **Visual Basic Editor** (Alt + F11)
- Affichez l'**Explorateur d'objets**



- Choisissez d'afficher les objets de votre classeur (VBAProject)

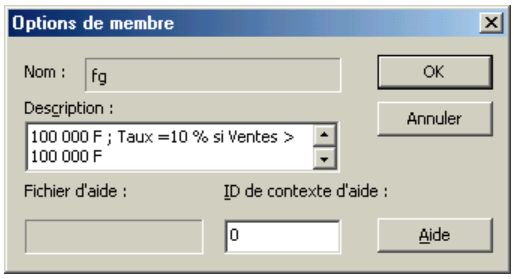


Les trois fonctions personnalisées que vous avez créées apparaissent :



- Faites un clic droit sur la fonction **fg** et choisissez la commande **Propriétés…**

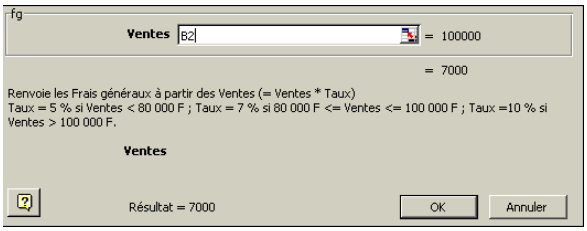
- Dans la zone Description : saisissez le texte qui apparaîtra dans l'Assistant Fonction
  Par exemple :
  Renvoie les Frais généraux à partir des Ventes (= Ventes * Taux) [CTRL]+[ENTREE]
  Taux = 5 % si Ventes < 80 000 F ; Taux = 7 % si 80 000 F <= Ventes <= 100 000 F ; Taux =10 % si Ventes > 100 000 F



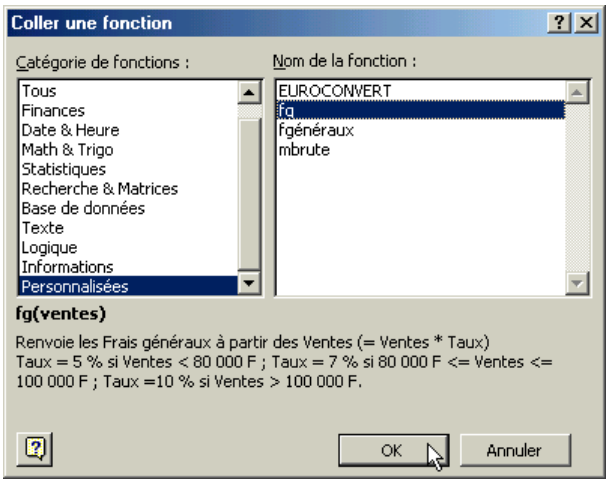
- Fermez la fenêtre **Explorateur d'objets**
- Fermez la fenêtre **Module1** et toutes les autres.
- Quittez **Visual Basic Editor**

Remarques :

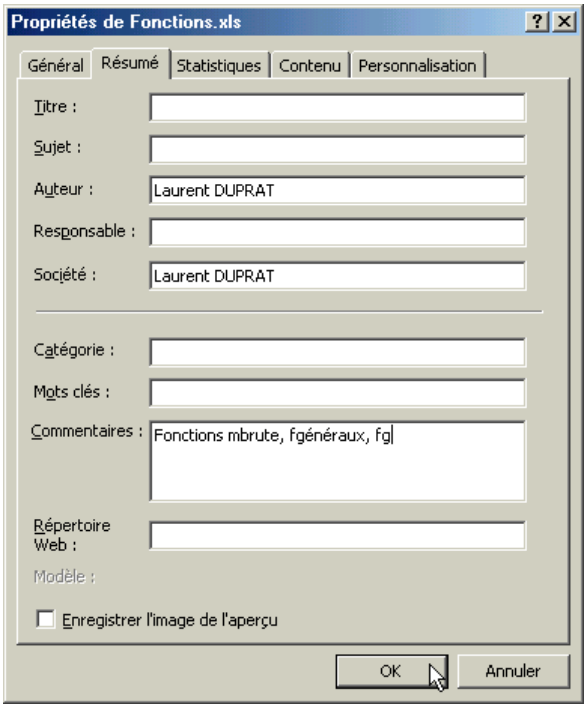
Vous pouvez visualiser la description saisie en cliquant dans la cellule contenant la fonction (ici, la cellule **B2**) puis en cliquant sur le bouton **Assistant Fonction** 



Le message est identique lorsque vous collez la fonction dans une cellule vide.



Vous pouvez créer un commentaire pour les deux autres fonctions si vous le désirez.

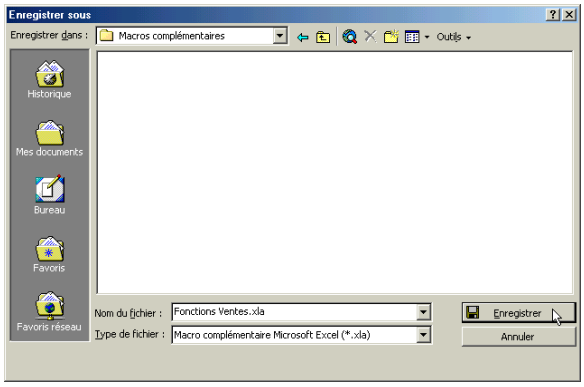


Un commentaire plus général peut être saisi dans le fichier Excel.

Depuis Excel,

- **Fichier**
- **Propriétés**
- Saisissez le commentaire souhaité dans la zone Commentaires de l'onglet **Résumé**.

3. 3. *Enregistrement de la macro complémentaire*



- **Fichier**
- **Enregistrer sous…**

- Choisissez comme type de fichier : **Macro complémentaire Microsoft Excel**.

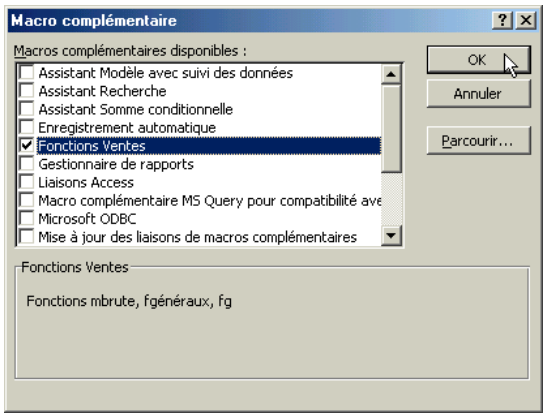
Excel place automatiquement ce fichier dans le dossier **C:\WINDOWS\Application Data\Microsoft\Macros complémentaires**

- Changez, éventuellement le nom : **Fonctions Ventes**.

Remarque : le chemin par défaut est utilisé pour un enregistrement sur votre disque dur (en local). Vous pouvez enregistrer un fichier Macro complémentaire (.xla) sur un serveur.

- Fermez le fichier **Fonctions**.

3. 4. Activation de la macro complémentaire



- Créez un nouveau classeur
- **Outils**
- **Macros complémentaires…**
- Activez la case de la macro complémentaire désirée (ici, la macro **Fonctions Ventes**)

Vos fonctions personnalisées sont désormais utilisables de n'importe quel classeur.

Remarques :
Notez la présence du message dans cette fenêtre.
Cliquez sur **Parcourir…** si le fichier .xla est enregistré ailleurs que dans le dossier **Macros complémentaires** de votre disque dur.

3. 5. Utilisation d'une fonction personnalisée complémentaire

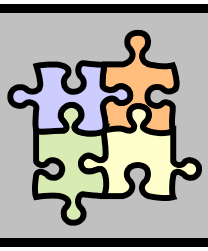
- Dans le nouveau classeur, saisissez l'exemple ci-contre.
- Dans la cellule **B3**, saisissez **=fg(B2)**
- Validez avec Entrée.

	A	B
1		Lannemezan
2	Ventes	55000
3	Frais généraux	2750
4		

- Fermez le fichier.

Page volontairement vide

LES MACROS



1. ENREGISTRER UNE MACRO

Nous allons concevoir une macro qui permet d'afficher automatiquement un titre que nous utilisons régulièrement. Ce titre comporte un texte et la date du jour. Il est encadré et centré sur 7 colonnes et 2 lignes.

1. 1. Définition des actions

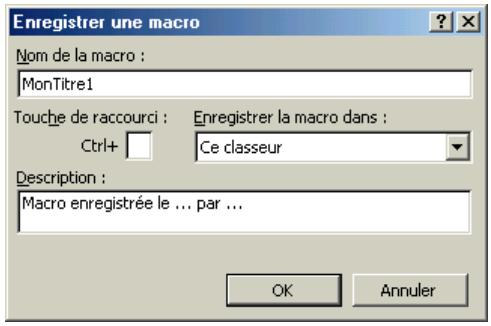
Pour créer ce titre, vous réalisez les actions suivantes :

- Saisie du texte dans une cellule
- Saisie de la date du jour dans la cellule du dessous
- Sélection d'une plage de 7 colonnes et 2 lignes
- Centrage des titres dans cette plage
- Encadrement de cette plage
- Annulation de la sélection en cours

C'est cette suite d'actions que nous allons enregistrer dans une macro.

1. 2. Enregistrement de la macro

- Lancez Excel
- **Outils**
- **Macro**
- **Nouvelle macro…**
- Saisissez **MonTitre1** comme nom de macro
- Cliquez sur **OK** pour lancer l'enregistrement de la macro



Excel affiche une barre d'outils contenant les boutons :

- Arrêter l'enregistrement
- Références relatives

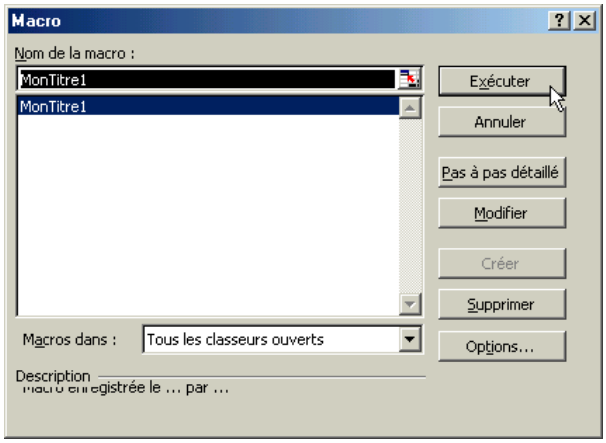
A partir de maintenant, toutes les actions que vous faites sont enregistrées dans la macro, y compris les mauvaises manipulations.

- En **A1** saisissez SITUATION AU
- Cliquez en **A2** et saisissez la formule =AUJOURDHUI()
- Sélectionnez la plage **A1:G2**
- **Format Cellule…**
  - **Alignement** : Centré sur plusieurs colonnes (horizontal)
  - **Bordure** : Contour
  - **OK**
- Cliquez en **A4** pour désactiver la sélection
- **Outils Options Affichage** : désactivez la case Quadrillage **OK**
- Cliquez sur le bouton **Arrêter l'enregistrement** 
- Enregistrez le classeur sous le nom **Macros**.

2. EXÉCUTER UNE MACRO

Nous allons tester la macro dans une nouvelle feuille.

- Activez la feuille **Feuil2** et cliquez en **A1**
- **Outils**
- **Macro**
- **Macros…**
- Choisissez la macro **MonTitre1**
- **Exécuter**

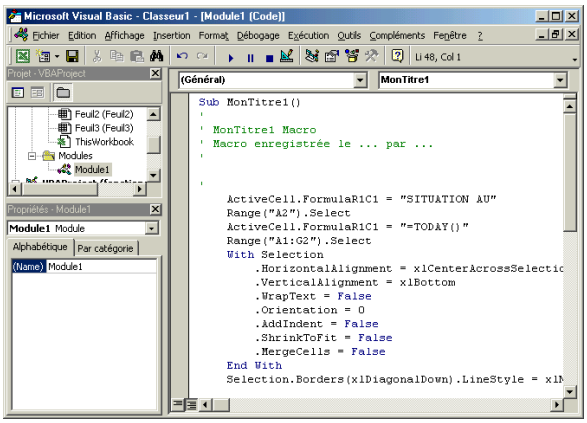


La macro s'exécute et lorsque l'exécution est terminée, la feuille **Feuil2** est semblable à la feuille **Feuil1**.

3. EXAMINER LE CODE DE LA MACRO

Excel a écrit la macro dans un module de l'**éditeur Visual Basic**.

- Affichez l'éditeur VBA avec le raccourci clavier Alt + F11
- Affichez éventuellement la fenêtre du **Module1** du **Classeur1**



La suite d'actions réalisées pendant l'enregistrement de la macro a été traduite en langage VBA qui utilise un vocabulaire et une syntaxe spécifiques.

Les lignes précédées d'une apostrophe et de couleur verte sont des commentaires.
Les mots de couleur bleue sont des mots clés.
Chaque ligne constitue une instruction, c'est-à-dire une action précise qu'Excel doit exécuter.

Nous allons analyser le contenu de cette macro.

Remarque : vous pouvez imprimer le **Module1**.

3. 1. Procédure

Pour définir le début et la fin de cette procédure, le code VBA utilise les mots clés **Sub** en début de macro et **End Sub** en fin de macro. Le nom de la procédure est celui de la macro suivi des parenthèses ouvrante et fermante.

```
Sub MonTitre1()
        Instructions
End Sub
```

3. 2. Instructions

Chaque ligne de la macro correspond à une des actions réalisées. Il existe différents types d'instructions : de sélection, d'affectation, de contrôle…

Sélection d'une cellule ou d'une plage de cellule

Voici les exemples de sélections rencontrées dans le code :

```
Range("A2").Select
Range("A1:G2").Select
Range("A4").Select
```

Range = Plage
Select = Sélectionner

La syntaxe générale du code VBA place les objets qui vont recevoir l'action avant la nature de l'action à effectuer. Ainsi pour comprendre une instruction, il suffit de la lire de droite à gauche : sélectionner la cellule A2.

Affectation d'un résultat

Lorsque les actions que vous faites entraînent des résultats, le code VBA fait alors appel à des instructions d'affectation. Elles se distinguent des précédentes car elles incluent un signe d'égalité qui signifie : affecter à.

Affecter une formule :

```
ActiveCell.FormulaR1C1 = "=TODAY()"
```

Cette instruction se lit : affecter la fonction AUJOURDHUI() à la cellule active.

Affecter une valeur :

```
ActiveCell.FormulaR1C1 = "SITUATION AU"
```

ActiveCell = Cellule active

Cette instruction se lit : affecter le texte SITUATION AU à la cellule active.

Remarque :
Vous pourriez remplacer ce code VBA par
ActiveCell.value = "SITUATION AU"

Affecter des propriétés :

Certaines instructions modifient les propriétés (ou caractéristiques) de l'objet sélectionné.

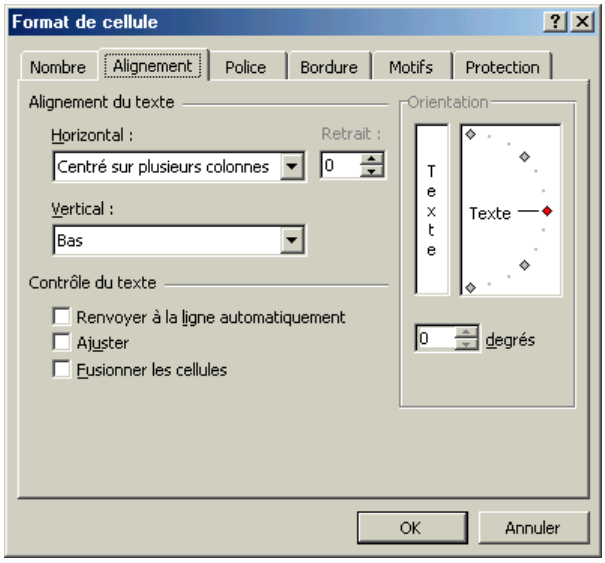
```
.HorizontalAlignment = xlCenterAcrossSelection
.WrapText = False
```

HorizontalAlignment = Alignement horizontal
xlCenterAcrossSelection = Centré sur plusieurs colonnes
WrapText = Renvoi à la ligne automatique

Ces deux instructions ont été écrites lorsque vous avez modifié les caractéristiques d'alignement des cellules sélectionnées :

Toute caractéristique (choisie dans une liste déroulante, une zone de saisie, une case d'option…) est précédée des lettres xl

Toute caractéristique correspondant à une case à cocher prend la valeur logique True ou False



Actions groupées

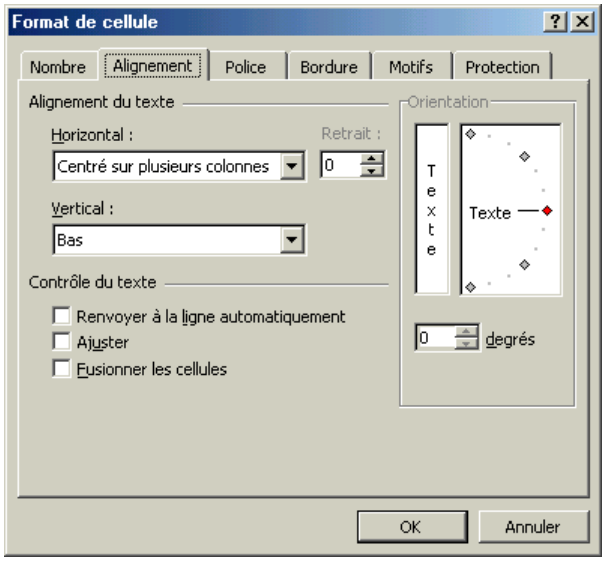
Lorsque plusieurs actions sont affectées à la même sélection, Visual Basic regroupe parfois ces actions à l'aide des mots clés **With** et **End With**.

```
With Selection
        .HorizontalAlignment = xlCenterAcrossSelection
        .VerticalAlignment = xlBottom
        .WrapText = False
        .Orientation = 0
        .ShrinkToFit = False
        .MergeCells = False
End With
```

4. MODIFIER LE CODE

4. 1. Supprimer le code inutile

Visual Basic semble avoir ajouté certaines actions. Lors de l'enregistrement, après avoir sélectionné la plage **A1:G2**, vous avez seulement modifié l'alignement horizontal. Visual Basic a ajouté une instruction par caractéristique se trouvant dans cet onglet.




```
With Selection
   .HorizontalAlignment = xlCenterAcrossSelection
   .VerticalAlignment = xlBottom
   .WrapText = False
   .Orientation = 0
   .ShrinkToFit = False
   .MergeCells = False
End With
```

Vous pouvez donc supprimer les lignes de code inutiles.

Ce qui donne :

```
With Selection
        .HorizontalAlignment = xlCenterAcrossSelection
End With
```

Ou encore :

```
Selection.HorizontalAlignment = xlCenterAcrossSelection
```

Remarque :
Si vous n'êtes pas sûr du code à supprimer, saisissez une apostrophe devant la ligne du code. Celle-ci sera transformée en commentaire et ne sera plus exécutée par Excel.

4. 2. Code utile

Une fois le code épuré, il reste ceci :

```
Sub MonTitre1()
' Macro enregistrée le ... par ...
    ActiveCell.FormulaR1C1 = "SITUATION AU"
    Range("A2").Select
    ActiveCell.FormulaR1C1 = "=TODAY()"
    Range("A1:G2").Select
    Selection.HorizontalAlignment = xlCenterAcrossSelection
    Selection.Borders(xlEdgeLeft).LineStyle = xlContinuous
    Selection.Borders(xlEdgeTop).LineStyle = xlContinuous
    Selection.Borders(xlEdgeBottom).LineStyle = xlContinuous
    Selection.Borders(xlEdgeRight).LineStyle = xlContinuous
    Range("A4").Select
    ActiveWindow.DisplayGridlines = False
End Sub
```

4. 3. Modifier une propriété

Nous désirons changer de style de bordure. Pour connaître les différentes possibilités, nous pouvons consulter l'aide de la propriété.

- Dans le code, double-cliquez sur la propriété **LineStyle** pour la sélectionner
- Appuyez sur la touche **F1**

L'aide nous indique les différentes possibilités.

- Fermez l'aide
- Remplacez la valeur **xlContinuous** par **xlDouble** pour les bordures du haut et du bas.

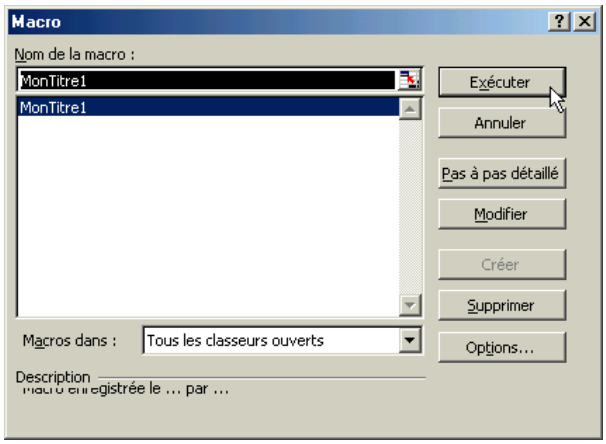
Ce qui donne :

```
    Selection.Borders(xlEdgeLeft).LineStyle = xlContinuous
    Selection.Borders(xlEdgeTop).LineStyle = xlDouble
    Selection.Borders(xlEdgeBottom).LineStyle = xlDouble
    Selection.Borders(xlEdgeRight).LineStyle = xlContinuous
```

5. EXÉCUTER UNE MACRO

Nous allons tester la macro modifiée dans une nouvelle feuille.

- Activez la feuille **Feuil3** et cliquez en **A4**
- **Outils**
- **Macro**
- **Macros…**
- Choisissez la macro **MonTitre1**
- **Exécuter**



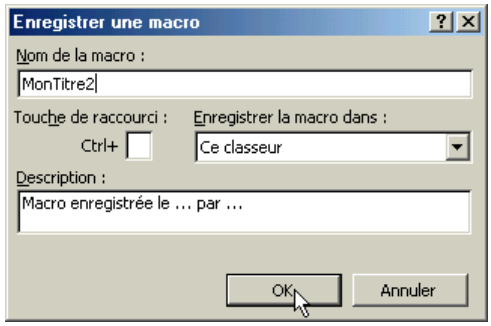
Le texte est bien dans la cellule **A4** (cellule active avant le lancement de la macro). La date se trouve dans la cellule **A2** et la mise en forme est bien appliquée à la plage **A1:G2**.

La macro **MonTitre1** fait toujours appel à des références absolues. Nous allons construire une macro identique en utilisant les références relatives pour pouvoir placer ce titre dans n'importe quel endroit de la feuille.

6. ENREGISTRER UNE MACRO AVEC DES RÉFÉRENCES RELATIVES

- Activez la feuille **Feuil4**.
- Cliquez dans la cellule **A1**.

- **Outils**
- **Macro**
- **Nouvelle macro…**
- Saisissez **MonTitre2** comme nom de macro
- Cliquez sur **OK** pour lancer l'enregistrement de la macro



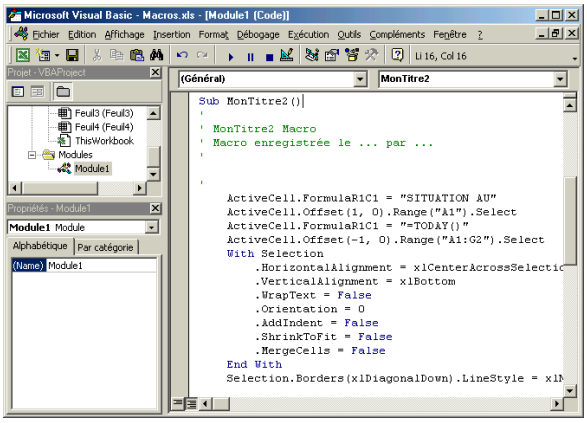
- Cliquez sur le bouton **Références relatives**  pour l'activer

- En **A1** saisissez SITUATION AU
- Cliquez en **A2** et saisissez la formule =AUJOURDHUI()
- Sélectionnez la plage **A1:G2**
- **Format Cellule…**
  - **Alignement** : Centré sur plusieurs colonnes (horizontal)
  - **Bordure** : Contour
  - **OK**
- Cliquez en **A4** pour désactiver la sélection
- **Outils Options Affichage** : désactivez la case Quadrillage **OK**
- Cliquez sur le bouton **Arrêter l'enregistrement** 

- Enregistrez le classeur.

7. EXAMINER LE CODE DE LA MACRO

- Affichez l'éditeur VBA avec le raccourci clavier Alt + F11
- Affichez éventuellement la fenêtre du **Module1** contenant le code de la macro **MonTitre2**



Toutes les instructions de la macro **MonTitre2** qui concernent les sélections de cellule ou de plage de cellules se présentent différemment :

```
ActiveCell.Offset(1, 0).Range("A1").Select
ActiveCell.Offset(-1, 0).Range("A1:G2").Select
ActiveCell.Offset(3, 0).Range("A1").Select
```

Offset(1, 0) = Décaler(lignes, colonnes)

Le premier exemple se lit de la façon suivante :
sélectionner une seule cellule à partir de la cellule placée au-dessous de la cellule active.

Le deuxième exemple se lit de la façon suivante :
sélectionner une plage de 7 colonnes et 2 lignes à partir de la cellule placée au-dessus de la cellule active.

Vous pouvez assainir le code pour qu'il ressemble à celui-ci :
(profitez-en pour renommer la macro)

```
Sub MonTitre()
    ActiveCell.FormulaR1C1 = "SITUATION AU"
    ActiveCell.Offset(1, 0).Range("A1").Select
    ActiveCell.FormulaR1C1 = "=TODAY()"
    ActiveCell.Offset(-1, 0).Range("A1:G2").Select
    Selection.HorizontalAlignment = xlCenterAcrossSelection
    Selection.Borders(xlEdgeLeft).LineStyle = xlContinuous
    Selection.Borders(xlEdgeTop).LineStyle = xlContinuous
    Selection.Borders(xlEdgeBottom).LineStyle = xlContinuous
    Selection.Borders(xlEdgeRight).LineStyle = xlContinuous
    ActiveCell.Offset(3, 0).Range("A1").Select
    ActiveWindow.DisplayGridlines = False
End Sub
```

- Fermez l'éditeur VBA

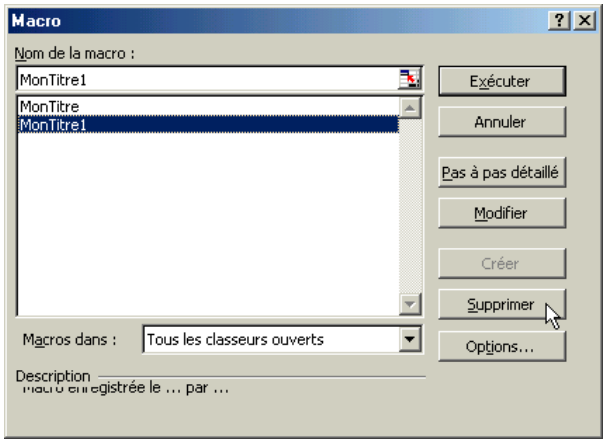
8. Exécuter une macro

- Testez la macro **MonTitre** à partir de la cellule **C7** dans la feuille **Feuil4**.

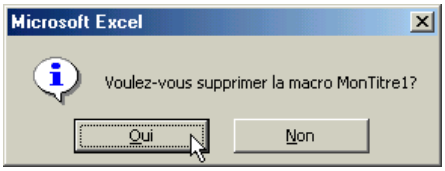
Le titre est correctement créé.

9. Supprimer une macro

- **Outils**
- **Macro**
- **Macros…**
- Sélectionnez **MonTitre1**
- Cliquez sur **Supprimer**



- Confirmez la suppression de la macro



10. RENDRE LA SAISIE INTERACTIVE.

Vous pouvez renseigner le titre au moment de l'exécution de la macro, c'est-à-dire, faire en sorte que la macro vous demande quel titre doit être pris en compte au-dessus de la date du jour.
Pour cela, nous allons utiliser la méthode qui affiche une boîte de dialogue prédéfinie permettant de saisir une information pendant l'exécution d'une macro.

```
InputBox("Message utilisateur", "Nom de la boîte")
```

InputBox = Boîte de dialogue Pour plus de détails sur cette méthode, consultez l'aide VBA.

- Si nécessaire, activez le **Module1**
- Supprimez les termes "SITUATION AU" dans le code de la macro
- A ce même endroit saisissez : InputBox("Tapez le titre souhaité :", "Saisie du titre")

De la même façon, vous pouvez avertir l'utilisateur que la macro est terminée.
Pour cela, nous allons utiliser la méthode qui affiche une boîte de message prédéfinie.

```
MsgBox "Message utilisateur", Type de bouton, "Nom de la boîte"
```

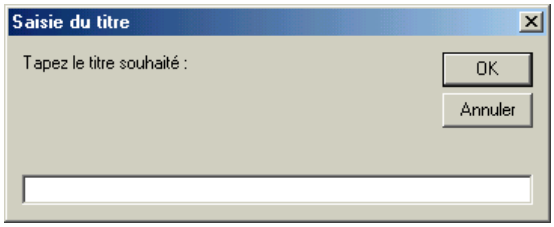
MsgBox = Boîte de message Pour plus de détails sur cette méthode, consultez l'aide VBA.

- Insérez en fin de macro la ligne d'instruction suivante :
  MsgBox "La macro est terminée.", vbOKOnly, "Fin de la macro"

- Fermez l'éditeur VBA

- Testez la macro **MonTitre** à partir de la cellule **A1** dans la feuille **Feuil4**.

Une première boîte apparaît :



- Saisissez le titre **Rapport annuel au**
- Cliquez sur **OK**

Remarque : si vous ne saisissez pas de titre ou si vous cliquez sur le bouton **Annuler**, la macro continue quand même à s'exécuter.

Une deuxième boîte apparaît :

11. OBLIGER LA SAISIE D'UN TITRE

L'utilisateur peut se servir de la première boîte de dialogue de façons différentes :
- il saisit le titre souhaité et clique sur OK,
- il saisit le titre souhaité et clique sur Annuler,
- il ne saisit pas de titre et clique sur OK,
- il ne saisit pas de titre et clique sur Annuler.

Nous pouvons contrôler la saisie et demander à l'utilisateur s'il souhaite saisir un titre ou pas. Pour cela, nous pouvons mettre en place la série d'instructions suivante :

Demander à l'utilisateur le titre qu'il souhaite saisir dans la cellule active	1
Si la cellule active est vide alors	2
Demander à l'utilisateur s'il souhaite saisir un titre (Réponse : Oui ou Non)	3
Fin Si	4
Si la Réponse est Oui	5
Tant que la cellule active est vide	6
Demander à l'utilisateur le titre qu'il souhaite saisir dans la cellule active	7
Fin Tant que	8
Fin Si	9

Deux nouveautés apparaissent dans cette suite d'instructions :
- stocker la réponse d'une boîte de message (MsgBox) pour tester cette réponse,
- une boucle Tant que.

Voici comment traduire ces instructions en VBA :

La ligne 3 peut se traduire par le code :

```
Réponse = MsgBox("Voulez-vous saisir un titre ?", vbYesNo, "Saisie du titre")
```

La variable **Réponse** contiendra la valeur du bouton sur lequel l'utilisateur a cliqué (dans notre exemple, Oui ou Non, c'est-à-dire en VBA vbYes ou vbNo)

La ligne 5 peut se traduire par le code :

```
If Réponse = vbYes Then
```

La ligne 6 peut se traduire par le code :

```
While ActiveCell.Value = ""
```

While indique le début de la boucle

La ligne 8 peut se traduire par le code :

```
Wend
```

Wend indique la fin de la boucle

Voici donc le code que vous pouvez saisir dans la macro **MonTitre**

```
Sub MonTitre()
' Macro enregistrée le ... par ...
   ActiveCell.FormulaR1C1 = InputBox("Tapez le titre souhaité :", "Saisie du titre")
   If ActiveCell.Value = "" Then
      Réponse = MsgBox("Voulez-vous saisir un titre ?", vbYesNo, "Saisie du titre")
   End If
   If Réponse = vbYes Then
      While ActiveCell.Value = ""
         ActiveCell.FormulaR1C1 = InputBox("Tapez le titre souhaité :", "Saisie du titre")
      Wend
   End If
…
```

- Testez la macro **MonTitre** à partir de la cellule **C7** dans la feuille **Feuil4**.

Le titre est correctement créé.

12. LES VARIABLES

Les variables permettent de stocker des valeurs intermédiaires. Elles contiennent tout type de données (**Entier**, **Long**, **Simple**, **Double**, **Monnaie**, **Booléen**, **Chaîne**, **DateHeure**).

Si vous utilisez une variable sans la déclarer (ce que nous avons fait avec la variable Réponse), elle prend le type **Variant**. Dans ce type particulier, c'est Excel qui détermine le type en fonction du contenu.

Pour déclarer une variable, il faut utiliser la syntaxe suivante (en début de code généralement) :

```
Dim Nom_de_la_variable As Type_de_Variable
```

Pour plus d'informations, consultez les rubriques d'aide VBA « Déclaration de variables » et « Portée et visibilité ».

13. LES INSTRUCTIONS DE CONTRÔLE (RAPPELS ET COMPLÉMENTS)

13. 1. Instructions de décision

Les instructions de décision permettent de faire des choix à l'intérieur d'une procédure

If … Then … Else

	En VBA

Si *condition* **Alors**
instructions
Sinon
instructions
Fin Si
	If *condition* **Then**
	instructions
	Else
	instructions
	End If

	En VBA

Si *condition1* **Alors**
instructions
SinonSi *condition2* **Alors**
Instructions
Sinon
instructions
Fin Si
	If *condition1* **Then**
	instructions
	ElseIf *condition2* **Then**
	instructions
	Else
	instructions
	End If

Les instructions **Sinon** et **SinonSi** sont facultatives. On peut ajouter autant d'instructions **SinonSi** que nécessaire.

Select Case

	En VBA
Selon Cas *Variable* **Cas** *Valeur01* *Instructions* **Cas** *Valeur02, Valeur03* *Instructions* **Cas** *Valeur04* **à** *Valeur10* *Instructions* **Cas est** > *Valeur20* *Instructions* **Cas Sinon** *Instructions* **Fin Selon**	**Select Case** *Variable* **Case** *Valeur01* *Instructions* **Case** *Valeur02, Valeur03* *Instructions* **Case** *Valeur04* **To** *Valeur10* *Instructions* **Case Is** > *Valeur20* *Instructions* **Case Else** *Instructions* **End Select**

13. 2. Instructions de boucle

Les instructions de boucle permettent de répéter plusieurs fois un même groupe d'instructions.

Do … Loop

Répète un bloc d'instructions :

- aussi longtemps qu'une condition est vraie (True)

	En VBA
Faire Tant que *condition* *instructions* **Sortir** *instructions* **Boucle**	**Do While** *condition* *instructions* **Exit Do** *instructions* **Loop**

- ou jusqu'à ce qu'une condition devienne vraie (True).

	En VBA
Faire Jusque *condition* *instructions* **Sortir** *instructions* **Boucle**	**Do Until** *condition* *instructions* **Exit Do** *instructions* **Loop**

Remarques :
Vous pouvez placer autant d'instructions **Exit Do** que vous le souhaitez dans une instruction **Do** … **Loop**, afin de sortir de la boucle. Souvent utilisée dans l'évaluation d'une condition (notamment **If** … **Then**), l'instruction **Exit Do** passe la main à l'instruction qui suit immédiatement l'instruction **Loop**.

Si vous avez choisi **While** (Tant que) et que la condition est fausse au départ, la boucle n'est jamais exécutée.
De même, si vous avez choisi **Until** (Jusque) et que la condition est déjà vraie, la boucle n'est jamais exécutée.
Pour que la boucle soit exécutée au moins une fois, il faut placer **While** ou **Until** après l'instruction **Loop**.

Do *instructions* **Exit Do** *instructions* **Loop While** *condition*	**Do** *instructions* **Exit Do** *instructions* **Loop Until** *condition*

While … Wend

Exécute une série d'instructions dans une boucle aussi longtemps que la valeur d'une condition est vraie (True).

	En VBA
Tant que *condition* *instructions* **Fin Tant que**	**While** *condition* *instructions* **Wend**

Remarque :
Si la valeur de l'argument *condition* est vraie (True), toutes les *instructions* sont exécutées jusqu'à ce que l'instruction **Wend** soit rencontrée. Le contrôle retourne ensuite à l'instruction **While** et *condition* est de nouveau vérifié. Si *condition* est toujours vraie (True), le processus est répété. Si la valeur de *condition* n'est pas vraie (True), l'exécution reprend à partir de l'instruction qui suit l'instruction **Wend**.

For … Next

Répète un groupe d'instructions le nombre de fois indiqué.

	En VBA
Pour *x* = *début* **à** *fin* **Par Pas De** *incrément* *Instructions* **Sortir** *instructions* **Suivant** *x*	**For** *x* = *début* **To** *fin* **Step** *incrément* *Instructions* **Exit For** *instructions* **Next** *x*

Remarques :

incrément peut être positif ou négatif. Si aucune valeur n'est indiquée, *incrément* prend par défaut la valeur 1.

Une fois que toutes les instructions de la boucle ont été exécutées, *incrément* est ajouté à *x*. Alors, les instructions de la boucle sont de nouveau exécutées (selon le résultat du même test que celui effectué à la première exécution), ou le contrôle sort de la boucle et exécute l'instruction qui suit immédiatement **Next**.

Toute modification de la valeur de *x* à l'intérieur de la boucle risque de rendre la lecture et la correction des erreurs du programme plus difficiles.

Vous pouvez également placer des instructions **Exit For** pour quitter la boucle à tout moment. L'instruction **Exit For** est souvent placée après l'évaluation d'une condition (**If** … **Then**, par exemple) ; elle passe la main à l'instruction située immédiatement après l'instruction **Next**.

For Each … Next

Répète un groupe d'instructions pour chaque élément d'un tableau ou d'une collection.

	En VBA
Pour *élément* **dans** *groupe* *Instructions* **Sortir** *instructions* **Suivant** *élément*	**For Each** *élément* **In** *groupe* *Instructions* **Exit For** *instructions* **Next** *élément*

Elément : Variable utilisée pour être répétée sur tous les éléments d'une collection. La variable *élément* peut uniquement être une variable de type Variant, une variable objet générique ou toute variable objet spécifique.
Groupe : Nom d'une collection.
Remarques

Le bloc **For** ... **Each** entre dans la boucle si l'argument *groupe* contient au moins un *élément*. Une fois le bloc entré dans la boucle, toutes les instructions de cette dernière sont appliquées au premier *élément* de *groupe*. Si *groupe* comporte plusieurs *éléments*, la boucle continue de s'exécuter pour chaque *élément*. Une fois que tous les *éléments* de *groupe* ont été traités, la boucle est fermée et l'exécution se poursuit par l'instruction située après l'instruction **Next**.

Vous pouvez également placer des instructions **Exit For** pour quitter la boucle à tout moment. L'instruction **Exit For** est souvent placée après l'évaluation d'une condition (**If** … **Then**, par exemple) ; elle passe la main à l'instruction située immédiatement après l'instruction **Next**.

13. 3. Instructions groupées

Exécute une série d'instructions appliquées à un seul objet ou à un type défini par l'utilisateur.

	En VBA
Avec *objet* *Instructions* **Fin Avec**	**With** *objet* *Instructions* **End With**

Remarque :
L'instruction **With** permet d'appliquer une série d'instructions à l'objet indiqué, sans qualifier à chaque fois le nom de l'objet. Par exemple, pour modifier plusieurs propriétés d'un seul objet, placez les instructions d'affectation de propriétés dans la structure de contrôle **With** ; vous ne faites ainsi référence qu'une seule fois à l'objet, au lieu de le faire à chaque affectation de propriété.

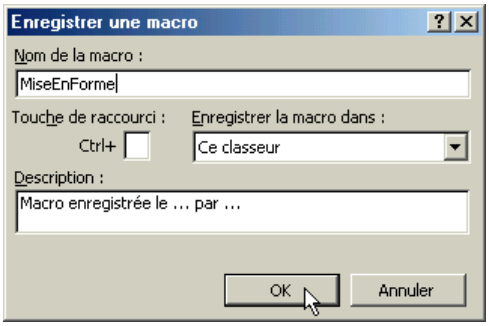
14. MACRO APPELANT UNE AUTRE MACRO

La macro **MonTitre** permet de saisir un titre et de l'encadrer. Nous n'avons pas prévu de modifier l'apparence des caractères de ce titre.
Il est possible de mémoriser dans une nouvelle macro la mise en forme souhaitée puis d'exécuter cette nouvelle macro depuis la macro **MonTitre**.

14. 1. Enregistrer une macro

- Activez la feuille **Feuil4**.
- Sélectionnez les cellules **A1** et **A2**.

- **Outils**
- **Macro**
- **Nouvelle macro…**
- Saisissez **MiseEnForme** comme nom de macro
- Cliquez sur **OK** pour lancer l'enregistrement de la macro

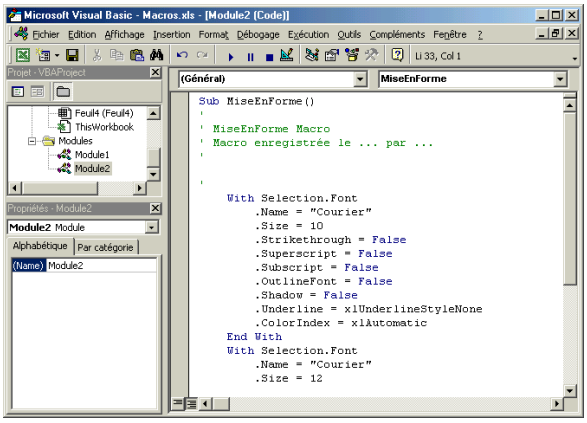


- Choisissez la police **Courier**
- Choisissez la taille **12**
- Cliquez sur le bouton **Gras** 
- Choisissez la couleur **Bleue** avec le bouton **Couleur de caractères** 
- Cliquez sur le bouton **Arrêter l'enregistrement** 

- Enregistrez le classeur.

14. 2. Examiner le code de la macro

- Affichez l'éditeur VBA avec le raccourci clavier Alt + F11
- Affichez éventuellement la fenêtre du **Module2** contenant le code de la macro **MiseEnForme**

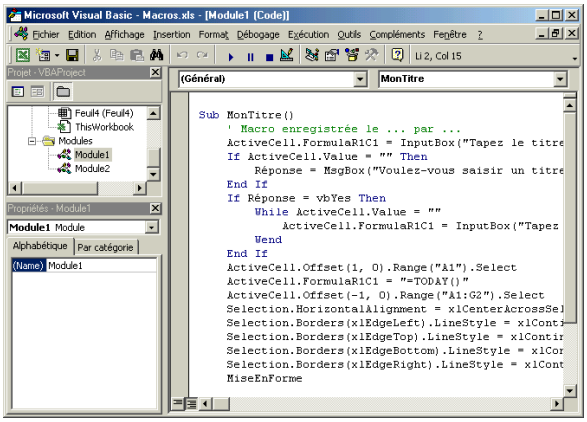


Vous pouvez assainir le code pour qu'il ressemble à celui-ci :

```
Sub MiseEnForme()
   With Selection.Font
      .Name = "Courier"
      .Size = 12
      .Bold = True
      .ColorIndex = 41
   End With
End Sub
```

14. 3. Appeler une macro dans une macro existante

- Affichez la fenêtre du **Module1** contenant le code de la macro **MonTitre**



Saisissez, dans une nouvelle ligne d'instructions, le nom de la macro à exécuter au moment désiré (ici, nous pouvons faire la mise en forme des caractères après la mise en forme des bordures) :

```
Selection.Borders(xlEdgeRight).LineStyle = xlContinuous
MiseEnForme
ActiveCell.Offset(3, 0).Range("A1").Select
```

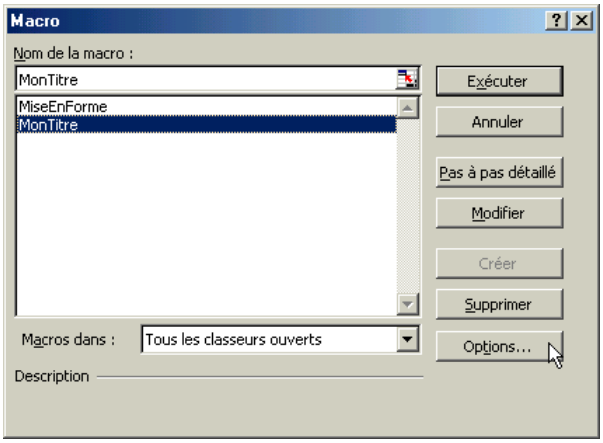
- Testez la macro **MonTitre** à partir de la cellule **C7** dans la feuille **Feuil4**.

Le titre est correctement créé.

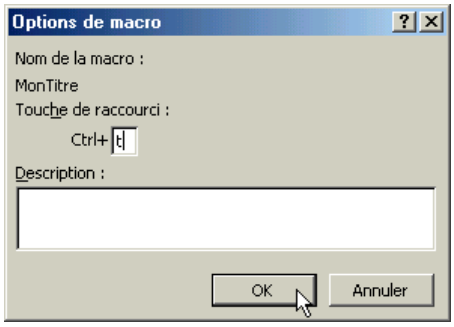
15. EXÉCUTER RAPIDEMENT UNE MACRO

15. 1. Affecter un raccourci clavier à une macro

- **Outils**
- **Macro**
- **Macros…**
- Sélectionnez la macro **MonTitre**
- Cliquez sur **Options…**



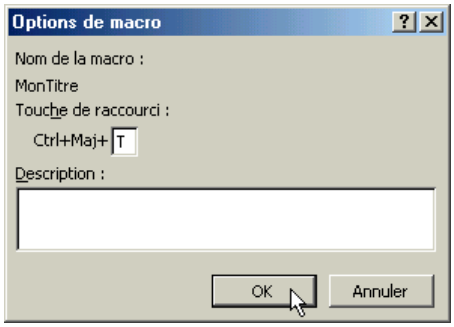
- Saisissez la touche de raccourci désirée (par exemple **t**)
- Cliquez sur **OK** pour valider l'option.
- Fermez la fenêtre Macro.



- Testez la macro **MonTitre** en utilisant son raccourci clavier **Ctrl + t** à partir de la cellule **C7** dans la feuille **Feuil4**.

Remarque :

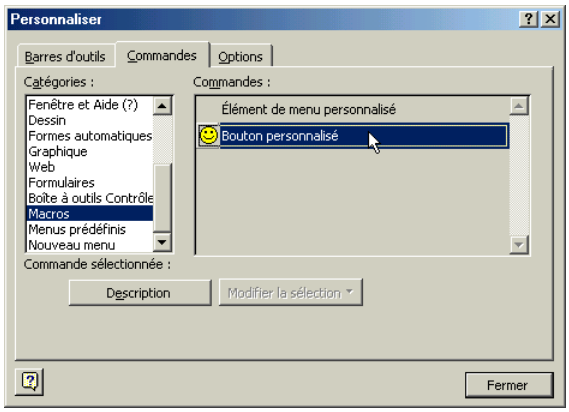
- Vous pouvez aussi utiliser la touche Majuscule Temporaire lors de la création d'un raccourci clavier.



15. 2. Affecter une macro à un bouton sur une barre d'outils

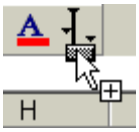
- **Affichage**
- **Barres d'outils…**
- **Personnaliser**

La fenêtre **Personnaliser** apparaît :



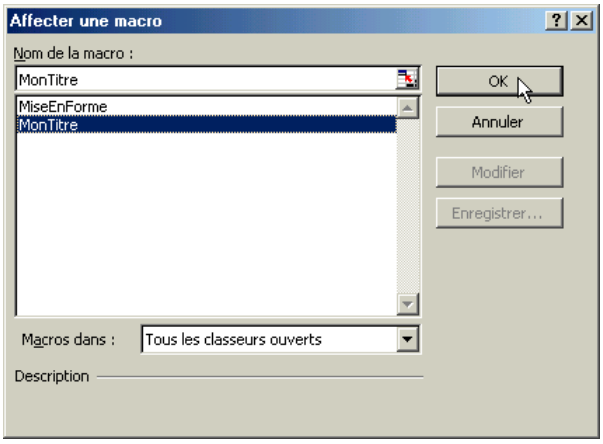
- Cliquez sur l'onglet **Commandes**
- Choisissez la catégorie **Macros**

- Cliquez-glissez **Bouton personnalisé** sur une barre d'outils



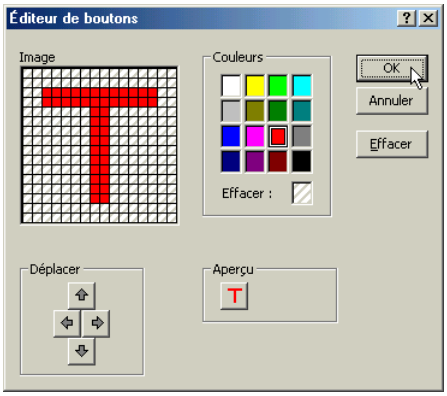
Le Bouton personnalisé est inséré sur la barre d'outils

- Cliquez avec le bouton droit sur le bouton personnalisé
- Cliquez sur **Affecter une macro**
- Choisissez la macro **MonTitre** puis **OK**

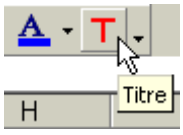


Dans le menu contextuel :

- choisissez **Editeur de bouton…** pour modifier son apparence.
- Saisissez le mot **Titre** dans la zone **Nom :**



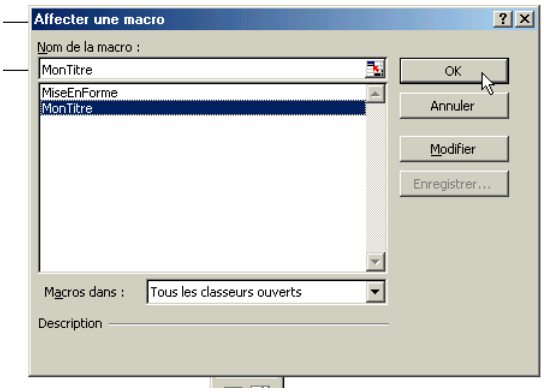
- Fermez la fenêtre **Personnaliser**.
- Cliquez dans de cellule **C7** dans la feuille **Feuil4**.
- Testez le bouton.



15. 3. Affecter une macro à un bouton sur une feuille

- **Affichage**
- **Barres d'outils…**
- **Formulaires**
- Dans la barre d'outils, choisissez l'outil **Bouton** 
- Tracez le bouton sur la feuille **Feuil4**

La fenêtre **Affecter une macro** apparaît :



- Choisissez **MonTitre**
- **OK**

- Cliquez dans le bouton et changez son nom et éventuellement la couleur du texte

- Fermez la barre d'outils **Formulaires**.
- Sélectionnez la cellule **C7**

- Pointer le bouton **Titre**.

Celui-ci est utilisable.

- Cliquez.

La macro s'exécute.

- Fermez et enregistrez le fichier.
- Quittez Excel.

16. DISPONIBILITÉ DES MACROS

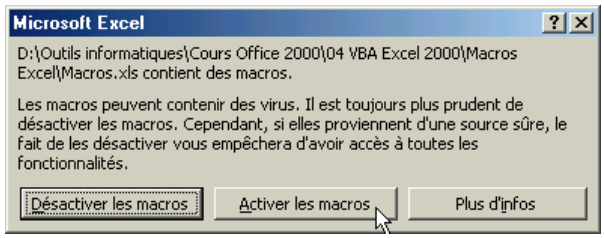
Telle qu'elle est créée, la macro **MonTitre** n'est disponible que dans le classeur **Macros**. Vous pouvez cependant l'utiliser dans un autre classeur, à condition que le classeur **Macros** soit ouvert.

- Lancez Excel.

Le bouton T est toujours présent puisque vous ne l'avez pas supprimé.

- Cliquez sur le bouton T

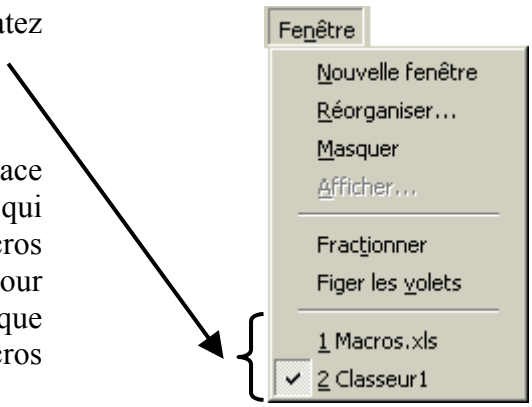
Excel ouvre le classeur **Macros**



- Activez les macros de ce classeur

La macro s'exécute.

- Cliquez sur le menu **Fenêtre** et constatez que les deux classeurs sont ouverts.



Remarque :
En utilisant le menu **Outils Macro** à la place du bouton, Excel ouvre la boîte **Macro** qui est vide et ne propose pas les macros existantes dans les autres classeurs. Pour qu'il les propose dans cette boîte, il faut que les classeurs qui contiennent les macros soient ouverts.

- Fermez les deux classeurs sans sauvegarder les modifications.

17. SUPPRIMER LE BOUTON SUR LA BARRE D'OUTILS

- Cliquez avec le bouton droit sur l'une des barres d'outils affichées.
- Cliquez sur **Personnaliser…**
- Faites glisser le bouton T dans cette boîte.
- Fermez cette boîte.

Le bouton est supprimé.

18. CLASSEUR DE MACROS PERSONNELLES

Afin de rendre les macros disponibles pour tous les classeurs, existants et à venir, vous devez enregistrer la macro dans le classeur de macros personnelles identifié sous le nom **PERSO.XLS**. Ce classeur n'existe pas tant que vous n'avez pas créé une macro qui doit y être stockée.

Il n'est pas forcément intéressant que toutes les macros soient écrites dans ce classeur. Seules les macros générales et essentielles doivent y être stockées.

Par exemple, nous allons créer une macro qui permet de protéger, dans une feuille, toutes les cellules qui contiennent des formules.

19. ENREGISTRER UNE MACRO DANS LE CLASSEUR PERSO.XLS

19. 1. Définition des actions

Rappel : Principe de la protection d'une feuille.
Toutes les cellules sont, par défaut, verrouillées mais cette propriété n'affecte pas les cellules tant que la feuille n'a pas été protégée.
Le verrouillage / déverrouillage des cellules se définit grâce à la boîte **Format de cellule**.
La protection / déprotection d'une feuille se définit à partir du menu **Outils Protection**.

Pour réaliser la protection, vous réalisez les actions suivantes :

- Sélection de toute les cellules de la feuille
- Déverrouillage de toutes les cellules
- Sélection de toutes les cellules qui stockent des formules
- Verrouillage de ces cellules
- Activation de la protection de la feuille

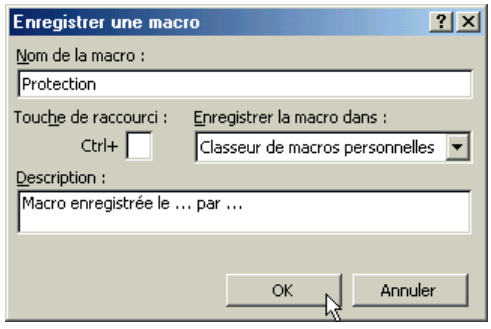
C'est cette suite d'actions que nous allons enregistrer dans une macro.

19. 2. Enregistrement de la macro

- Ouvrez le classeur **Macros** et activez la feuille **Feuil1**.

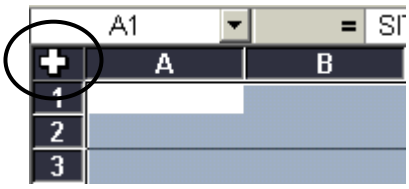
- Vérifiez que cette feuille contienne au moins une formule de calcul (la fonction **=AUJOURDHUI()**, par exemple).

- **Outils**
- **Macro**
- **Nouvelle macro…**
- Saisissez **Protection** comme nom de macro
- Choisissez Enregistrer la macro dans : **Classeur de macros personnelles**
- Cliquez sur **OK** pour lancer l'enregistrement de la macro



Sélection de toute les cellules de la feuille :

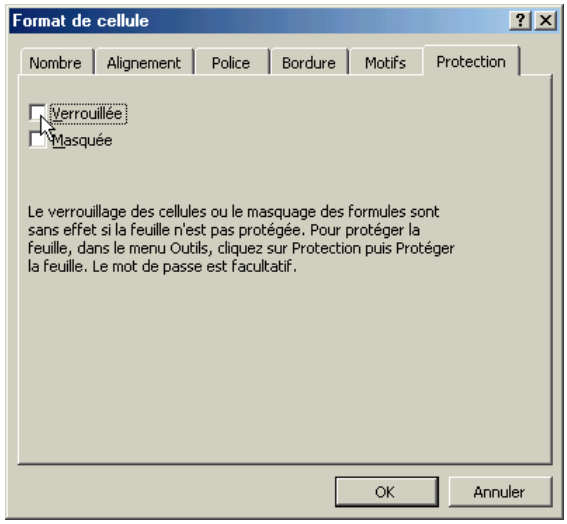
- Cliquez sur le rectangle à l'intersection des lignes et des colonnes.



Toutes les cellules de la feuille sont sélectionnées

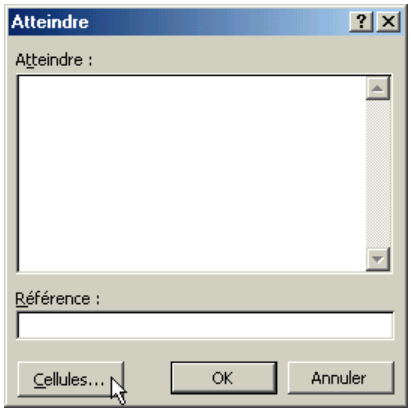
Déverrouillage de toutes les cellules :

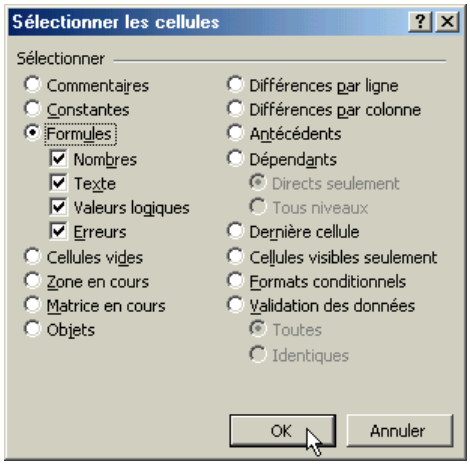
- **Format Cellule…Protection**
- Cliquez **Verrouillée** pour désactiver cette case
- **OK**



Sélection de toutes les cellules qui stockent des formules :

- **Edition atteindre…**
- Cliquez sur **Cellules…** pour choisir le type de cellules à sélectionner

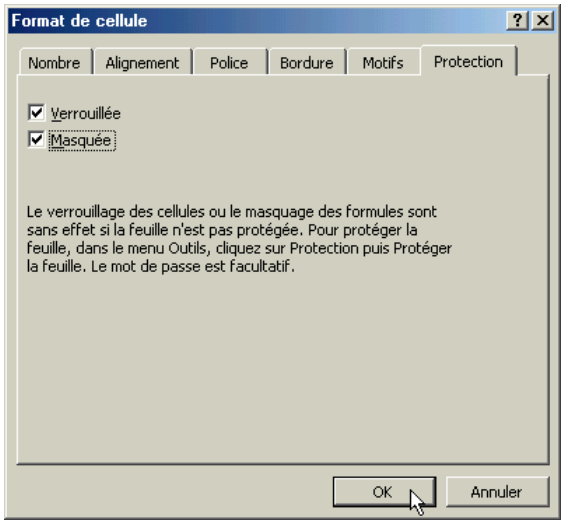




- Cliquez sur **Formules** pour ne sélectionner que les cellules de cette nature.
- **OK**

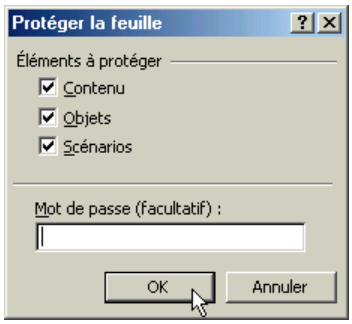
Dans la feuille, la seule cellule sélectionnée est la cellule **A2** contenant une formule.

Verrouillage de ces cellules :



- **Format Cellule…Protection**
- Cliquez **Verrouillée** pour activer cette case
- Cliquez sur **Masquée** pour activer cette case
- **OK**

Activation de la protection de la feuille :

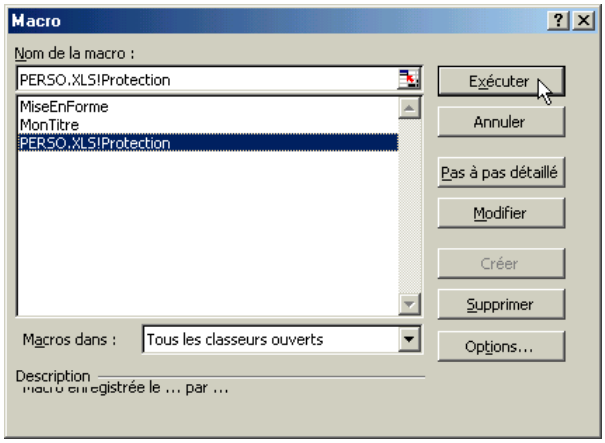


- **Outils Protection Protéger la feuille…**
- **OK**

- Cliquez sur le bouton **Arrêter l'enregistrement** 

19. 3. Exécuter une macro

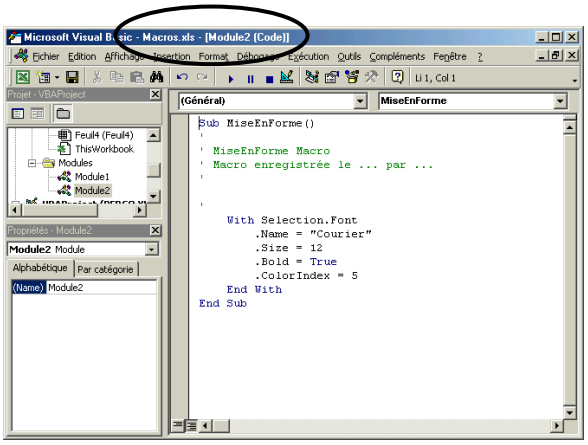
Nous allons tester la macro dans une autre feuille.



- Activez la feuille **Feuil2**
- **Outils**
- **Macro**
- **Macros…**
- Choisissez la macro **PERSO.XLS!Protection**
- **Exécuter**

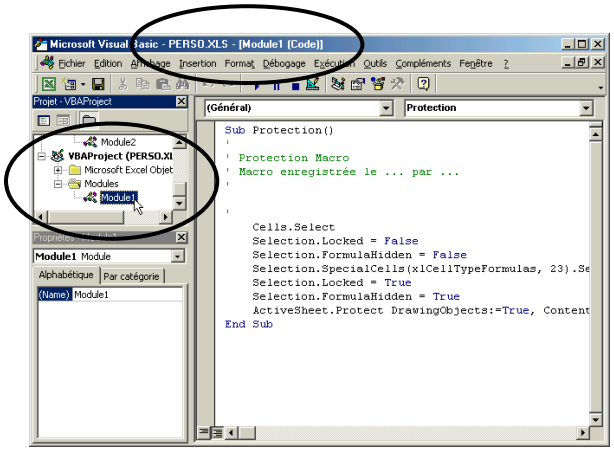
La macro s'exécute et lorsque l'exécution est terminée, les cellules contenant des calculs dans la feuille **Feuil2** sont protégées.

19. 4. Examiner le code de la macro



- Affichez l'éditeur VBA avec le raccourci clavier Alt + F11

L'éditeur VBA affiche un module du fichier **Macros**.



- Développez l'arborescence du classeur **PERSO.XLS** pour visualiser ses modules.
- Double-cliquez sur **Module1** pour l'afficher.

Le code de la macro **Protection** apparaît :

Voici le code :

```
Sub Protection()
'
' Protection Macro
' Macro enregistrée le ... par ...
'

'
    Cells.Select
    Selection.Locked = False
    Selection.FormulaHidden = False
    Selection.SpecialCells(xlCellTypeFormulas, 23).Select
    Selection.Locked = True
    Selection.FormulaHidden = True
    ActiveSheet.Protect DrawingObjects:=True, Contents:=True, Scenarios:=True
End Sub
```

19. 5. Modifier le code

Nous souhaitons que l'utilisateur de la macro choisisse ou non de masquer le contenu des cellules protégées (c'est-à-dire, dans notre exemple, les formules de calculs).

Pour cela, la macro peut poser la question à l'utilisateur et, en fonction de la réponse, la macro masque ou pas le contenu de ces cellules (instructions abordées page 32).

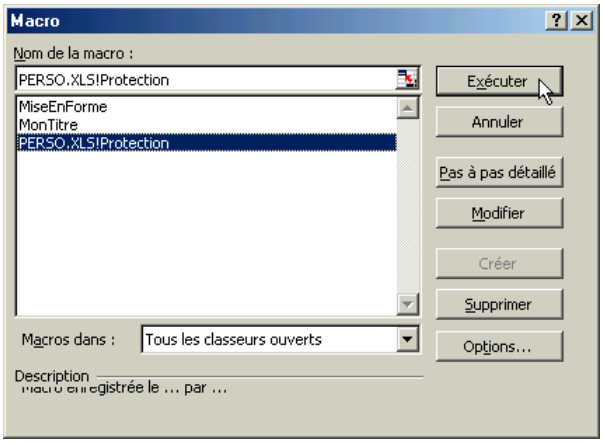
Vous pouvez modifier le code pour qu'il se présente de la façon suivante :

```
Sub Protection()
' Protection Macro
' Macro enregistrée le ... par ...
Cells.Select
    Selection.Locked = False
    Selection.FormulaHidden = False
    Selection.SpecialCells(xlCellTypeFormulas, 23).Select
    Selection.Locked = True
    Réponse = MsgBox("Voulez-vous masquer les formules de calcul ?", vbYesNo, _
    "Masquer les cellules")
    If Réponse = vbYes Then
        Selection.FormulaHidden = True
    Else
        Selection.FormulaHidden = False
    End If
    ActiveSheet.Protect DrawingObjects:=True, Contents:=True, Scenarios:=True
End Sub
```


19. 6. Exécuter une macro

Nous allons tester la macro modifiée dans une autre feuille.

- Fermez l'éditeur VBA



- Activez la feuille **Feuil3**
- **Outils**
- **Macro**
- **Macros…**
- Choisissez la macro **PERSO.XLS!Protection**
- **Exécuter**

La fenêtre **Masquer les cellules** apparaît :

- Cliquez sur le bouton de votre choix.

La feuille est protégée. Les cellules contenant des fonctions de calcul ne sont pas modifiables. Leur contenu est visible ou pas dans la barre de formule en fonction de votre choix.

20. OÙ SE TROUVE LE CLASSEUR PERSO.XLS ?

Avant de fermer le fichier **Macros**,

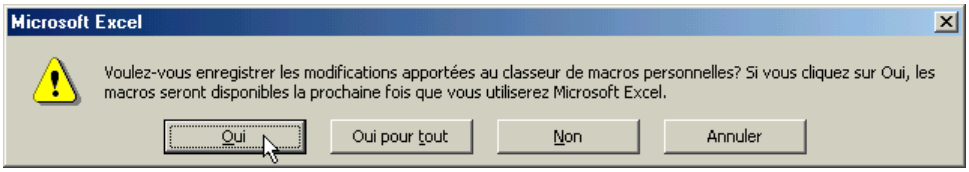
- Cliquez sur le menu **Fenêtre**

Seul le fichier **Macros** est ouvert.
Remarquez que de la commande **Afficher…** est désormais activée (elle ne l'était pas jusqu'à présent, pour le vérifier voir copie du menu page 43).

- Quittez **Excel**

Excel nous demande si nous souhaitons enregistrer les modifications apportées au classeur de macros personnelles.

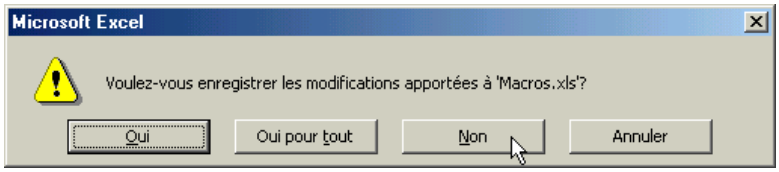
- Répondez par **Oui**, sous peine de perdre la macro **Protection** créée dans le classeur **PERSO.XLS** non enregistrée.



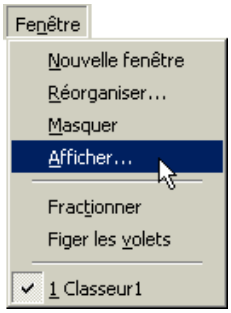
Excel nous demande si nous souhaitons enregistrer les modifications apportées au classeur **Macros**.

- Répondez par **Non**.

Les feuilles ne seront pas protégées.

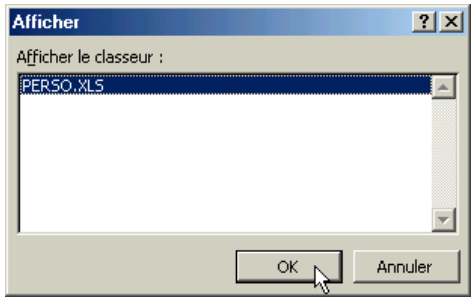


- Lancez **Excel**.

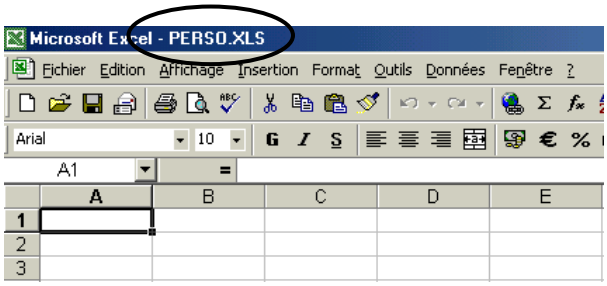


Un classeur Classeur1 est créé.

- Cliquez sur le menu **Fenêtre**
- Cliquez sur la commande **Afficher…**

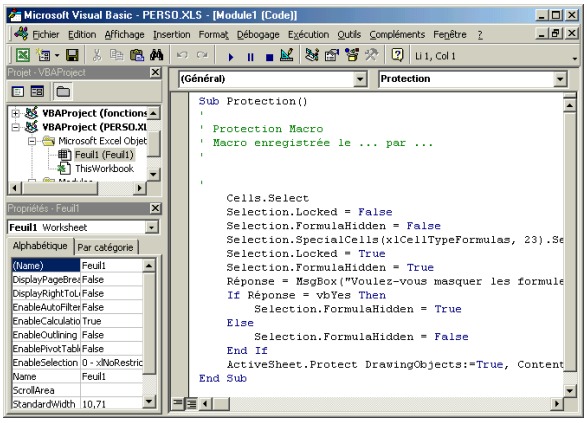


- Sélectionnez le classeur **PERSO**
- Cliquez sur **OK**



Le classeur **PERSO** est affiché.
Celui-ci ne contient qu'une feuille de calcul.

- Affichez l'éditeur VBA



Nous retrouvons le module **Module1** contenant la macro **Protection** que vous pourriez éventuellement modifier.

- Fermez l'éditeur VBA

- Cliquez sur le menu **Fenêtre**
- Cliquez sur la commande **Masquer**

- Quittez **Excel**
- Enregistrez les modifications apportées au fichier de macros personnelles.

Le fichier **PERSO.XLS** est, par défaut, toujours masqué. Stocké dans le dossier de démarrage **XLSTART** (chemin : **C:\WINDOWS\Application Data\Microsoft\Excel\XLSTART**), celui-ci est automatiquement ouvert mais masqué à chaque fois que vous chargez Excel (c'est pourquoi les macros qui y sont stockées sont disponibles en permanence).

Une **MACRO** copiée-collée d'un classeur quelconque dans un module VBA du classeur **PERSO.XLS** sera accessible et utilisable depuis n'importe quel classeur.

Une **FONCTION** copiée-collée d'un classeur quelconque dans un module VBA du classeur **PERSO.XLS** sera accessible et utilisable depuis n'importe quel classeur.

Fin du support de notes.