VBA

Introduction

Visual Basic for Application (VBA) est un environnement de programmation qui accompagne et permet d'automatiser la plupart des applications bureautiques de Microsoft.

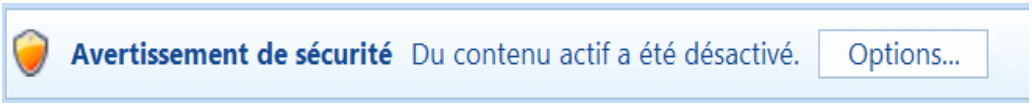
Connaissances préalables

La compréhension de ce qui suit requiert une connaissance fonctionnelle de Excel.

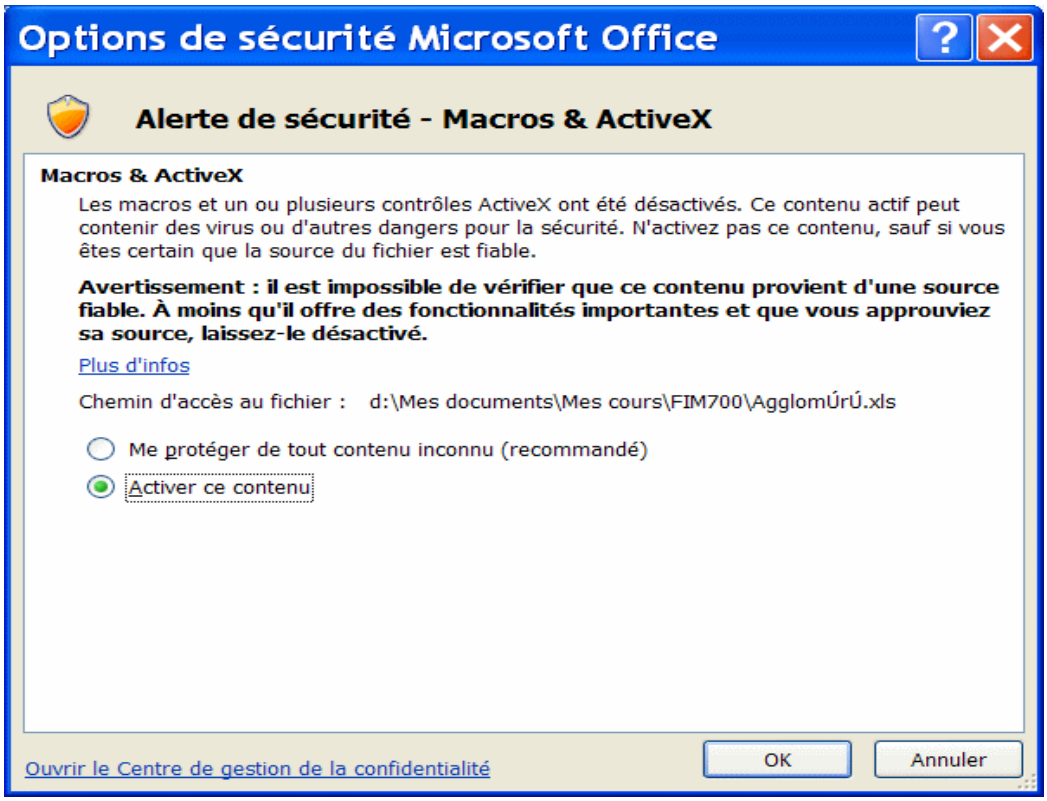
Vous pourrez améliorer vous connaissances sur ce logiciels en lisant le contenu des sections sur Excel 2003 et Excel 2007.

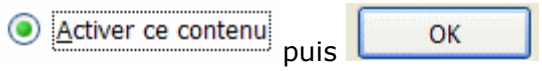
Excel et la sécurité

Excel refuse souvent d'exécuter les programmes VBA (qu'il nomme aussi macro-commandes ou macros). À l'ouverture d'un classeur contenant du code VBA, l'avertissement suivant apparaît sous le ruban:



Appuyez sur le bouton Options...:



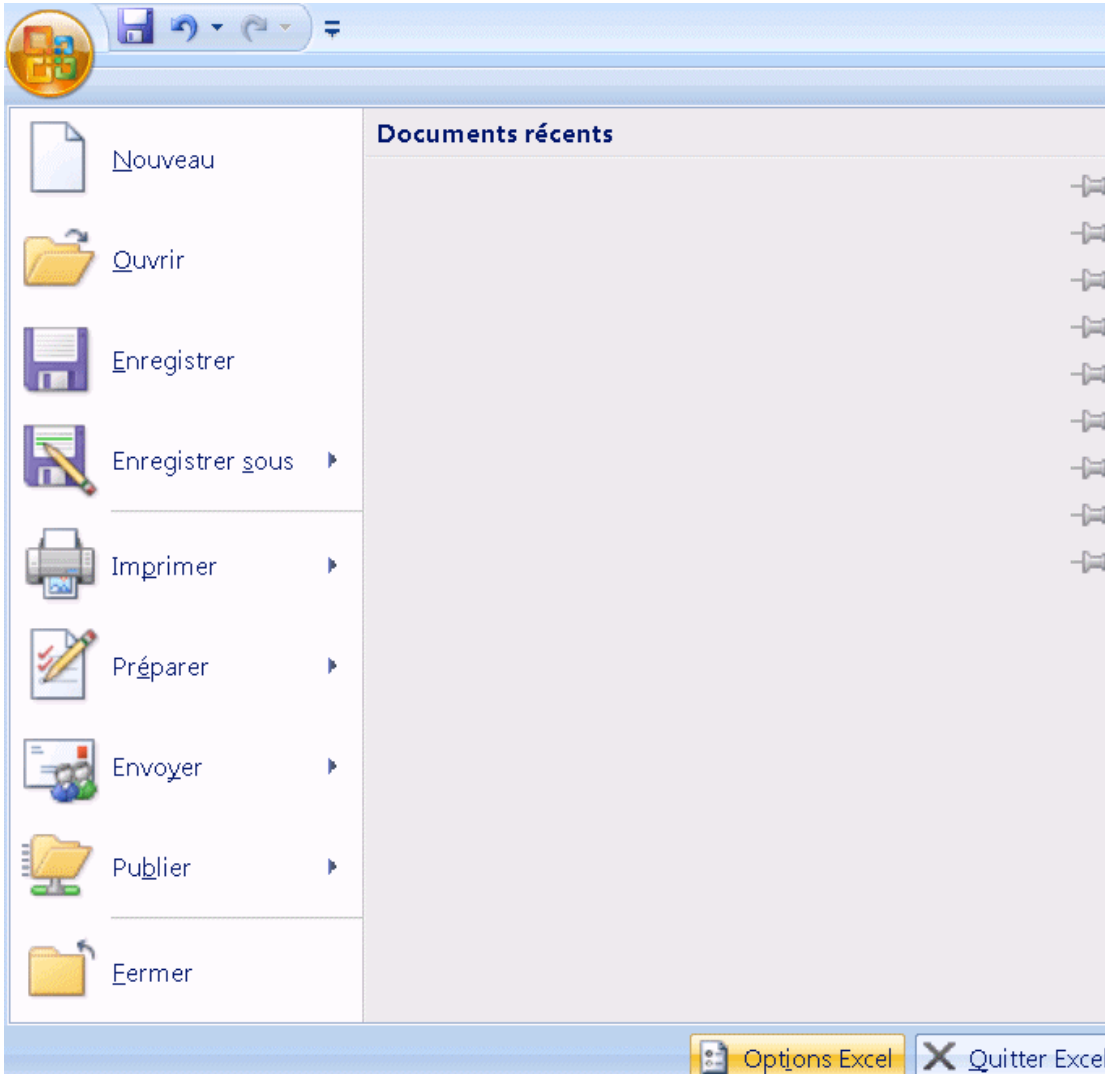
Sélectionnez 

puis

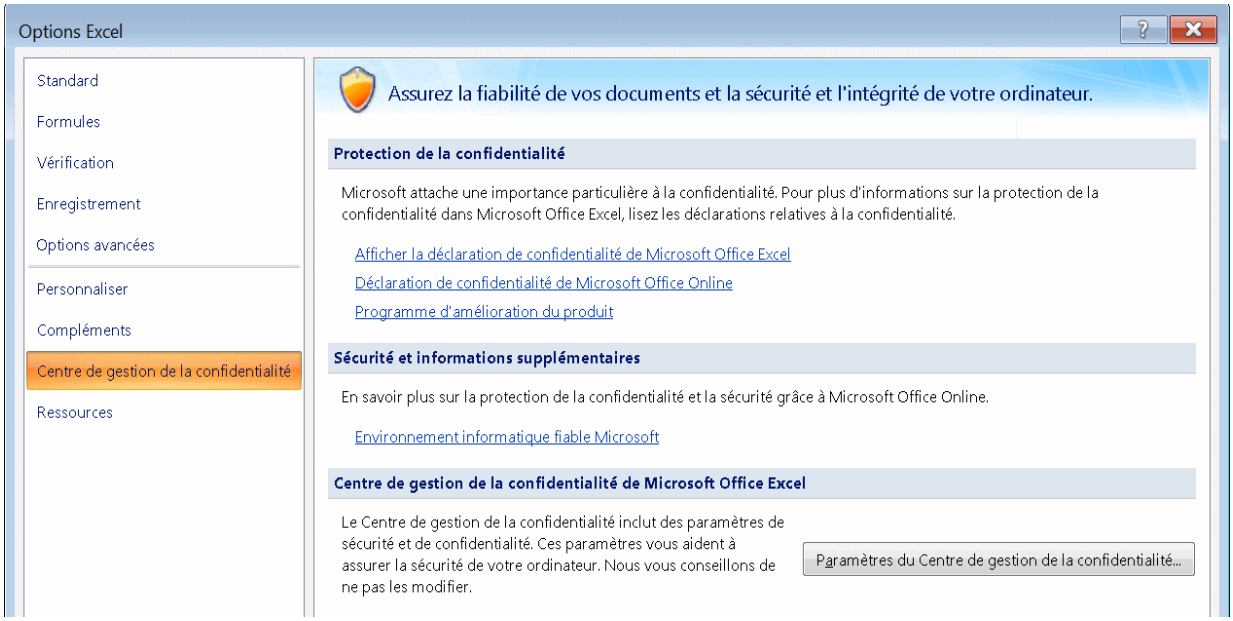
Il est aussi possible que VBA soit complètement désactivé sur votre ordinateur.

Dans Excel 2007:

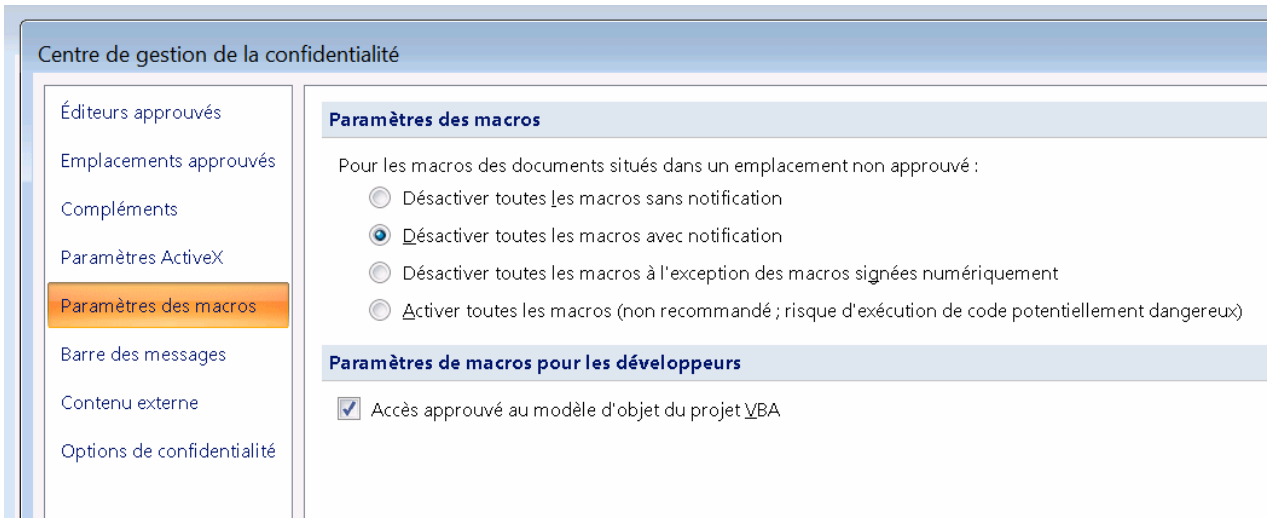
Cliquez le bouton Office  .



Cliquez le bouton **Options Excel**.



- Sélectionnez le **Centre de gestion de la confidentialité**.
- Cliquez le bouton **Paramètres du Centre de gestion de la confidentialité.**

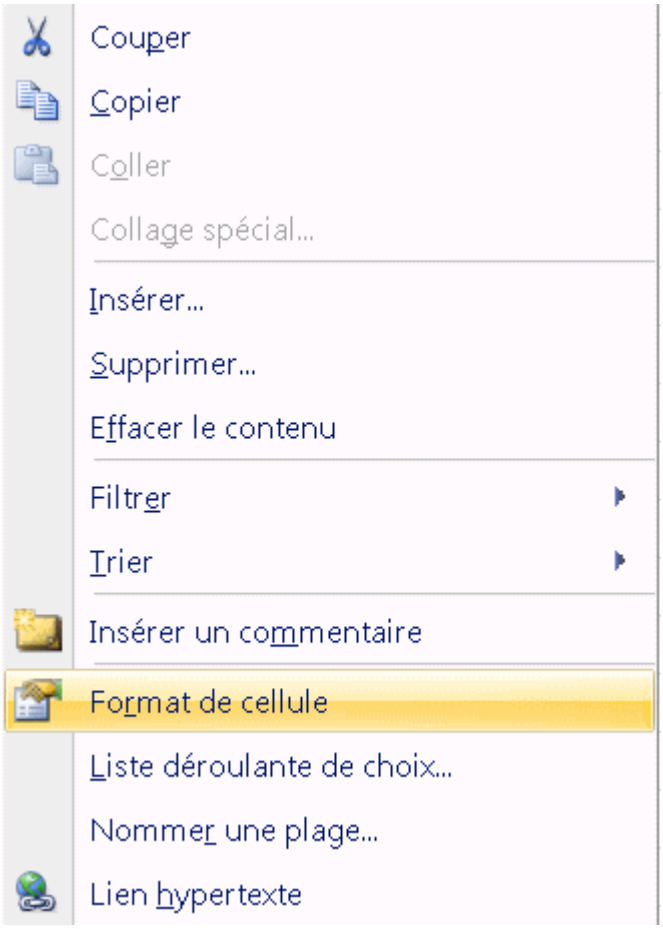


- Sélectionnez l'option **Paramètres des macros**.
- Sélectionnez le choix **Désactiver toutes les macros avec notification**.
- Appuyez sur le bouton **OK**.

Vous devez quitter Excel pour que le changement à cette option soit activé.

Premier exemple: Identifier par une couleur les cellules déverrouillées de la feuille de travail active

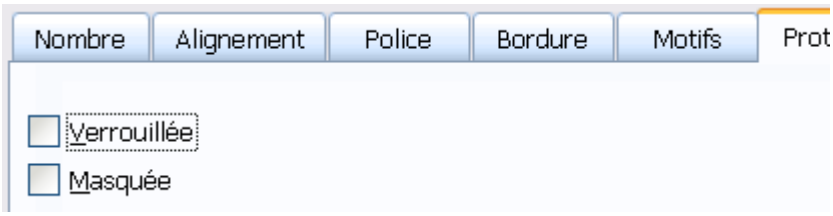
Ce premier exemple illustre l'utilisation de VBA pour automatiser une tâche Excel qu'il serait fastidieuse de faire manuellement:
colorer en jaune chaque cellule déverrouillée.



- Démarrez Excel avec un classeur vierge.
- Sélectionne le cellules **A2**, **B1** et **D1**.

Gardez undoigt sur la touche **Ctrl** afin de pouvoir sélectionnez plusieurs cellules éloignées les unes des autres.

- Placez le pointeur par-dessus la cellule **A2**.
- Appuyez sur le bouton **droit** de la souris.
- De la liste des options du menu contextuel, sélectionnez l'option **Format de cellule.**



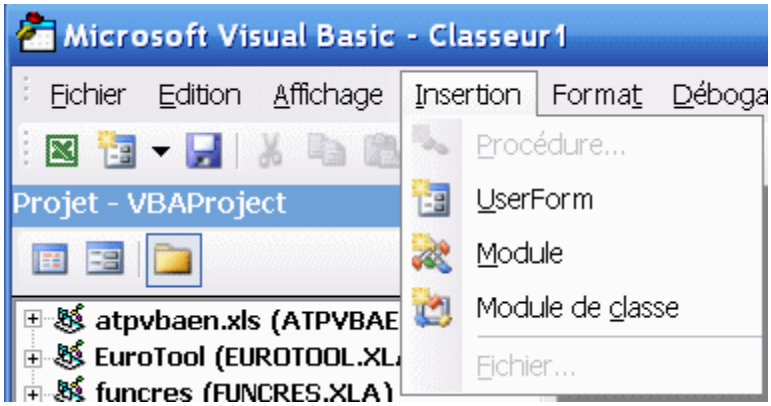
- Sélectionnez l'onglet **Protection**.
- Désactivez la case **Verrouillée**.
- Répétez pour les cellules **B1** et **D1**.

- Ouvrez l'onglet développeur à droite:

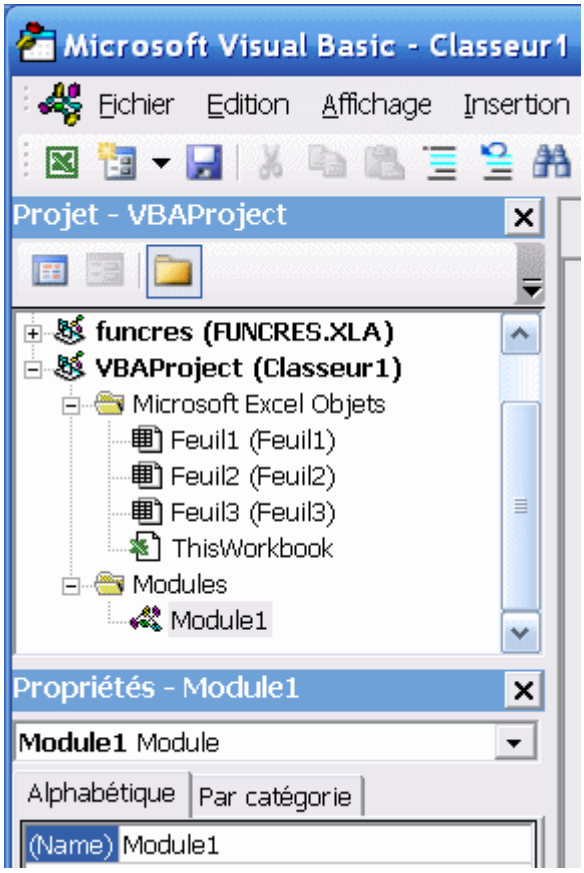
S'il n'est pas affiché:

- Appuyez sur le bouton Office  .
- Appuyez sur le bouton **Options Excel** (en bas à droite).

- Dans le menu Standard, sélectionnez la 3e case à cocher: **Afficher l'onglet Développeur** dans le ruban.
- Appuyez sur le bouton **Visual Basic**.



- Du menu **Insertion**, sélectionnez **module**:



- Changez le nom du module dans la zone de texte (**Name**) en bas à gauche. Remplacez**Module1** par un nom significatif ne contenant pas d'espace **ApprentissageVBA** par exemple.

```
Sub ArrièrePlan()
'Auteur: Michel Berthiaume
'Mettre en jaune les cellules non protégées de la feuille
active

Dim rCellule As Range

For Each rCellule In ActiveSheet.Cells
    If Not rCellule.Locked Then
        rCellule.Interior.Color = vbYellow
    End If
```

- Tapez (ou copiez-collez!) le code VBA suivant dans la zone de texte à droite.

Cela devrait donner le résultat ci-dessous.

```
    Application.StatusBar = "Traitement de la cellule " &
rCellule.Address
Next
End Sub
```

(Général) ArrièrePlan

```
Option Explicit
Sub ArrièrePlan()
'Auteur: Michel Berthiaume
'Mettre en jaune les cellules non protégées de la feuille active

Dim rCellule As Range

For Each rCellule In ActiveSheet.Cells
    If Not rCellule.Locked Then
        rCellule.Interior.Color = vbYellow
    End If
    Application.StatusBar = "Traitement de la cellule " & rCellule.Address
Next
End Sub
```

Testez votre code:

- Placez le curseur sur la première ligne (**SUB**).
- Appuyez sur la touche **F8**.

La ligne se colore en jaune.

```
Sub ArrièrePlan()
'Auteur: Michel Berthiaume
'Mettre en jaune les cellules non protégées de la feuille active

Dim rCellule As Range

For Each rCellule In ActiveSheet.Cells
    If Not rCellule.Locked Then
        rCellule.Interior.Color = vbYellow
    End If
    Application.StatusBar = "Traitement de la cellule " & rCellule.Address
Next
End Sub
```

VBA exécute une ligne à chaque fois que vous enfoncez la touche **F8**, en répétant les instructions encadrées par For et Next
pour chaque cellule de la feuille Excel active.

```
For Each rCellule In ActiveSheet.Cells
    If Not rCellule.Locked Then
        rCellule.Interior.Color = vbYellow
    End If
    Application.StatusBar = "Traitement de la cellule " & rCellule.Address
Next
End Sub
```

Vérifiez les changements dans la feuille Excel. Remarquez que la barre d'état d'Excel indique l'adresse de la cellule traitée par VBA.

Pour terminer l'exécution du programme, cliquez sur les boutons.

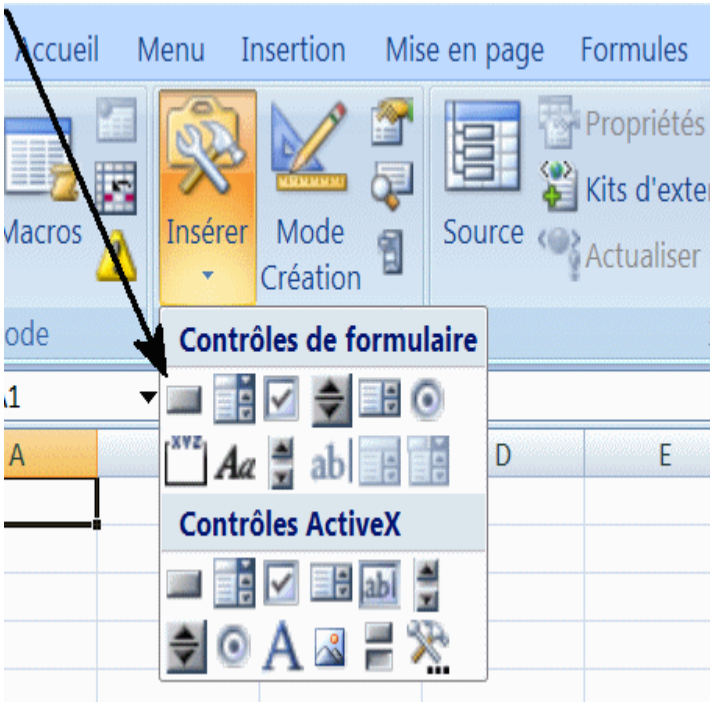
 **Continuer**

Le programme s'exécutera pour chaque cellule. Si le sablier tarde à disparaître, arrêtez le programme en enfonçant ensembles les touches **Ctrl-Pause/Arrêt** (ou **Break**).

ou

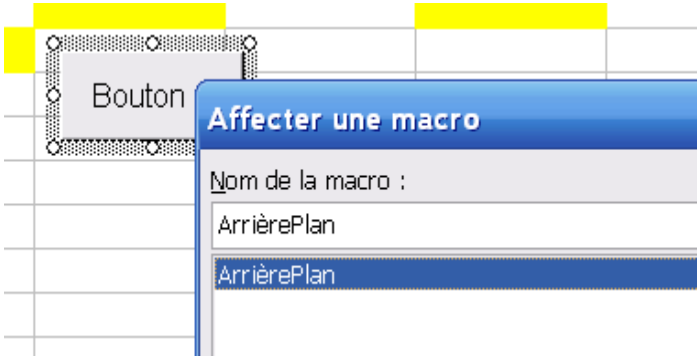
 **Réinitialiser**: le programme s'arrêtera.

ATTENTION: vous devez terminer l'exécution d'un programme VBA selon la procédure ci-dessus avant d'en exécuter un autre. En effet, si un programme est en débogage, VBA est en pause.



Rendez le programme disponible en Excel.

Retournez dans la feuille Excel, onglet **développeur**.
Dans le ruban **développeur**, bloc **Insérer Contrôle**, sélectionnez le Contrôle de formulaire **Bouton**.



Dessinez un bouton dans la feuille active.
Sélectionnez **ArrièrePlan** dans la liste.

Verrouillez quelques cellules jaunes, déverrouillez-en d'autres.
Cliquez sur le bouton pour que les cellules déverrouillées se colorent en jaune.

Quelques commentaires sur ce programme:

♦Il illustre l'un des 3 types de programme qu'on peut réaliser avec VBA: une procédure SUB.
♦Il contient plusieurs catégories d'instructions:

- Deux commentaires: 'Auteur: Michel Berthiaume et 'Mettre en jaune les cellules non protégées de la feuille active
- Une déclaration: Dim rCellule As Range
- Une boucle: For Each rCellule In ActiveSheet.Cells ... Next.
- Un test: If ... Then ... End If.
- Deux assignations: rCellule.Interior.Color = vbYellow et Application.StatusBar = "Traitement de la cellule " & rCellule.Address.
- Une variable contenant un objet (une cellule Excel): rCellule.
- Une constante VBA: vbYellow.
- Un objet Excel (la barre d'état): Application.StatusBar.
- Une constante littérale: "Traitement de la cellule ".
- Deux opérateurs: Not et &.
- Trois propriétés: Color (de l'objet rCellule.Interior), locked et Address (de rCellule).

♦Il contient aussi une maladresse de programmation: une cellule jaune qui est verrouillée reste jaune après avoir été déverrouillée.

Second exemple: Compter le nombre de cellules d'une couleur donnée dans une plage donnée

Dans ce second exemple, on veut pouvoir écrire dans une cellule Excel la formule: **=fnNbCellulesCouleur(A1:D3,D1)** et ainsi afficher dans cette cellule le nombre de cellules de la plage A1:D3 dont la couleur est identique à la couleur de la cellule D1. Évidemment, on veut pouvoir remplacer A1:D3 et D1 par toute autre référence à une plage ou une cellule valide.

Démarrez Excel avec un classeur vierge (ou utilisez le classeur créé pour l'exemple précédent).
Colorez l'arrière-plan de quelques cellules.
Démarrez l'éditeur VBA.
Créez un module ou utilisez un module existant.

```
Function fnNbCellulesCouleur(Plage As Range, Couleur As
Range) As Long
'Auteur: Michel Berthiaume
'Compter le nombre de cellules d'une couleur donnée dans
une plage donnée
'Plage: plage de cellules à inspecter
'Couleur: cellule de la couleur cherchée

Dim rCellule As Range

For Each rCellule In Plage
    If rCellule.Interior.Color = Couleur.Interior.Color
Then
```

Tapez (ou copiez-collez!) le code VBA suivant dans la zone de texte à droite.

Le résultat devrait être comme l'image ci-dessous.

```
        fnNbCellulesCouleur = fnNbCellulesCouleur + 1
    End If
Next
End Function

Sub Test()
Dim t

    t = fnNbCellulesCouleur(Range("A1:D3"), Range("D1"))

End Sub
```

```
Function fnNbCellulesCouleur(Plage As Range, Couleur As Range) As Long
'Auteur: Michel Berthiaume
'Compter le nombre de cellules d'une couleur donnée dans une plage donnée
'Plage: plage de cellules à inspecter
'Couleur: cellule de la couleur cherchée

Dim rCellule As Range

For Each rCellule In Plage
    If rCellule.Interior.Color = Couleur.Interior.Color Then
        fnNbCellulesCouleur = fnNbCellulesCouleur + 1
    End If
Next
End Function
```

```
Sub Test()
Dim t

    t = fnNbCellulesCouleur(Range("A1:D3"), Range("D1"))

End Sub
```

```
Function fnNbCellulesCouleur(Plage As Range, Couleur As Range) As Long
'Auteur: Michel Berthiaume
'Compter le nombre de cellules d'une couleur donnée dans une plage donnée
'Plage: plage de cellules à inspecter
'Couleur: cellule de la couleur cherchée

Dim rCellule As Range

For Each rCellule In Plage
    If rCellule.Interior.Color = Couleur.Interior.Color Then
        fnNbCellulesCouleur = fnNbCellulesCouleur + 1
    End If
Next
End Function
```

```
Sub Test()
Dim t

    t = fnNbCellulesCouleur(Range("A1:D3"), Range("D1"))

End Sub
```

```vba
Function fnNbCellulesCouleur(Plage As Range, Couleur As Range) As Long
'Auteur: Michel Berthiaume
'Compter le nombre de cellules d'une couleur donnée dans une plage donnée
'Plage: plage de cellules à inspecter
'Couleur: cellule de la couleur cherchée

Dim rCellule As Range

For Each rCellule In Plage
    If rCellule.Interior.Color = Couleur.Interior.Color Then
        fnNbCellulesCouleur = fnNbCellulesCouleur + 1
    End If
Next
End Function

Sub Test()
Dim t

    t = fnNbCellulesCouleur(Range("A1:D3"), Range("D1"))

End Sub
```

VBA exécute une ligne à chaque fois que vous enfoncez la touche **F8** D'abord la 3e ligne de **SUB TEST**.

```vba
Function fnNbCellulesCouleur(Plage As Range, Couleur As Range) As Long
'Auteur: Michel Berthiaume
'Compter le nombre de cellules d'une couleur donnée dans une plage donnée
'Plage: plage de cellules à inspecter
'Couleur: cellule de la couleur cherchée

Dim rCellule As Range

For Each rCellule In Plage
    If rCellule.Interior.Color = Couleur.Interior.Color Then
        fnNbCellulesCouleur = fnNbCellulesCouleur + 1
    End If
Next
End Function

Sub Test()
Dim t

    t = fnNbCellulesCouleur(Range("A1:D3"), Range("D1"))

End Sub
```

```
Function fnNbCellulesCouleur(Plage As Range, Couleur As Range) As Long
'Auteur: Michel Berthiaume
'Compter le nombre de cellules d'une couleur donnée dans une plage donnée
'Plage: plage de cellules à inspecter
'Couleur: cellule de la couleur cherchée

Dim rCellule As Range

For Each rCellule In Plage
    If rCellule.Interior.Color = Couleur.Interior.Color Then
        fnNbCellulesCouleur = fnNbCellulesCouleur + 1
    End If
Next
End Function

Sub Test()
Dim t

    t = fnNbCellulesCouleur(Range("A1:D3"), Range("D1"))

End Sub
```

- Appuyez sur la touche **F8** pour exécuter chaque ligne de la fonction.

- Pour terminer l'exécution de la fonction, cliquez sur les boutons.

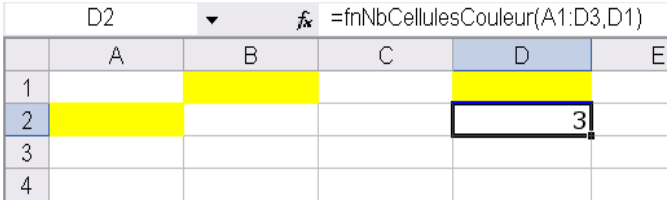
 **Continuer**: la fonction s'exécutera pour chaque cellule de la plage.
Si le sablier tarde à disparaître, arrêtez la fonction en enfonçant ensembles les touches **Ctrl-Pause/Arrêt** (ou **Break**).

OU

 **Réinitialiser**: la fonction s'arrêtera.

ATTENTION:
Vous devez terminer l'exécution d'une fonction VBA selon la procédure ci-dessus avant d'exécuter un autre programme VBA. En effet, si un programme est en débogage, VBA est en pause.

Utilisez la nouvelle fonction dans Excel.



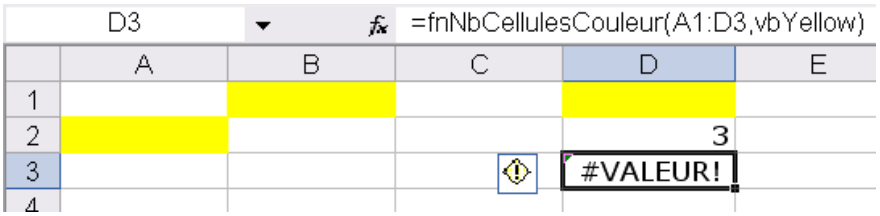
- Dans une feuille du classeur Excel, colorez l'arrière plan de quelques cellules (A2, B1 et D1 par exemple).
- Dans une cellule de la feuille, inscrire la formule**=fnNbCellulesCouleur(A1:D3,D1)** en remplaçant **A1:D3** par la référence à une plage qui contient les cellules à dénombrer, et **D1** par la référence à une cellule de la couleur voulue.

Quelques commentaires sur le programme:

♦Il illustre l'un des 3 types de programme qu'on peut réaliser avec VBA: une fonction personnalisée.
♦Il contient plusieurs catégories d'instructions:

- Plusieurs commentaires: les lignes commençant par '.
- Deux paramètres de fonction: Plage et Couleur.
- Quatre déclarations: fnNbCellulesCouleur(...) As Long, Plage As Range, Couleur As Range, rCellule As Range
- Une boucle: For Each rCellule In Plage ... Next.
- Un test: If ... Then ... End If.
- Une expression logique: rCellule.Interior.Color = Couleur.Interior.Color
- Une assignation: fnNbCellulesCouleur = fnNbCellulesCouleur + 1.
- Une constante littérale: 1.
- Un opérateur: +.
- Une variable contenant un objet (une cellule Excel): rCellule.
- Une propriété: Color.

♦Il contient aussi une maladresse de programmation: si l'utilisateur entre des paramètres erronés (**=fnNbCellulesCouleur(A1:D3,vbYellow** ou **=fnNbCellulesCouleur(A1:D3)** par exemple), le message d'erreur n'est pas explicite.
♦Il contient aussi une procédure Sub (Sub Test) qui sert à tester la fonction VBA à partir de l'environnement VBA.

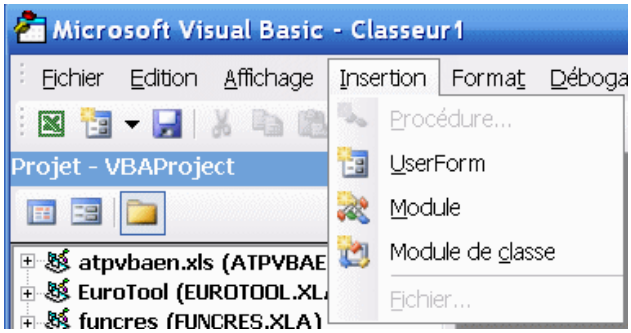


Troisième exemple: Boîte de dialogue d'accueil

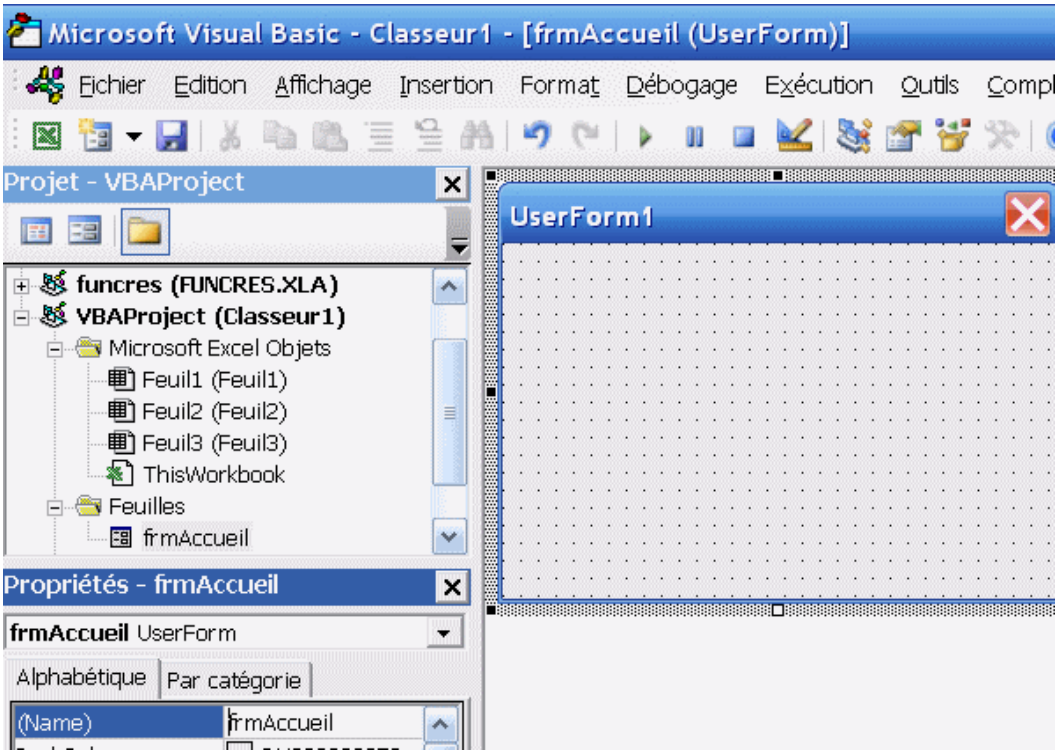
Dans ce troisième exemple, on veut qu'une fenêtre d'accueil s'affiche automatiquement lors de l'ouverture d'un classeur Excel.

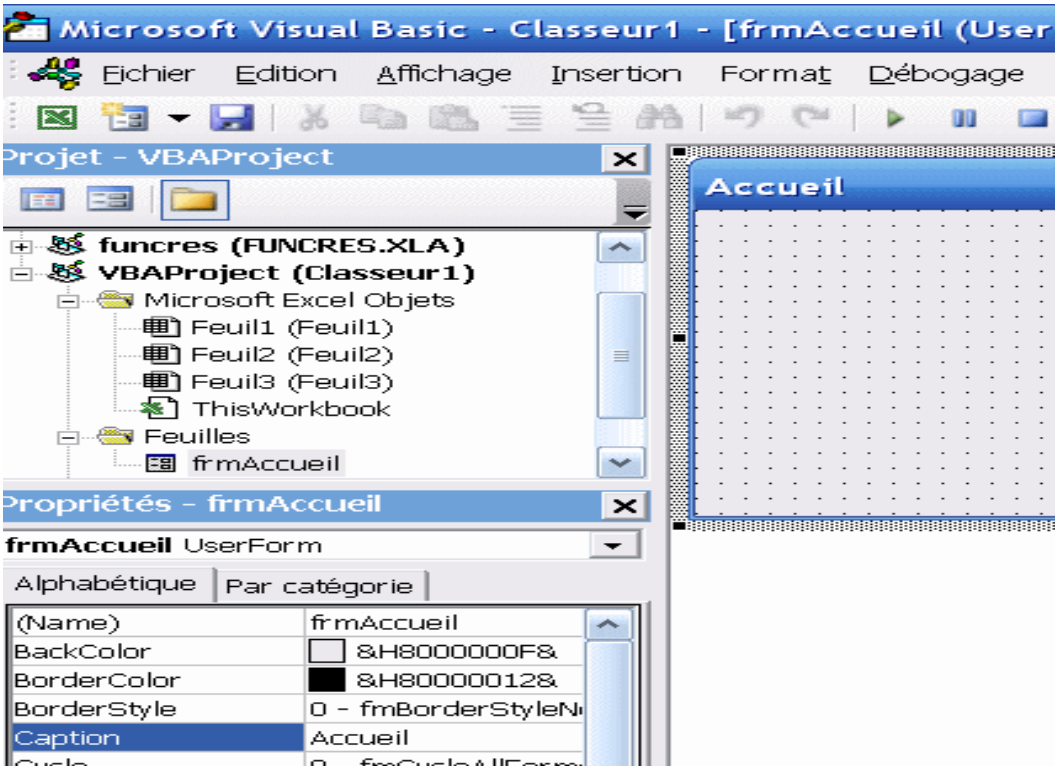
●Démarrez Excel et ouvrez un classeur (vierge ou non).
● Démarrez l'éditeur VBA.

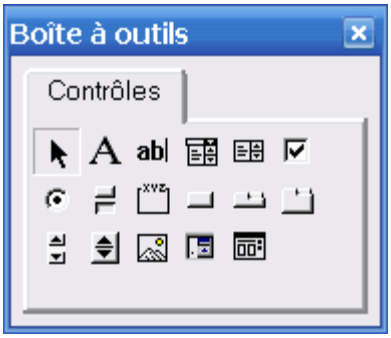
Créez un formulaire:

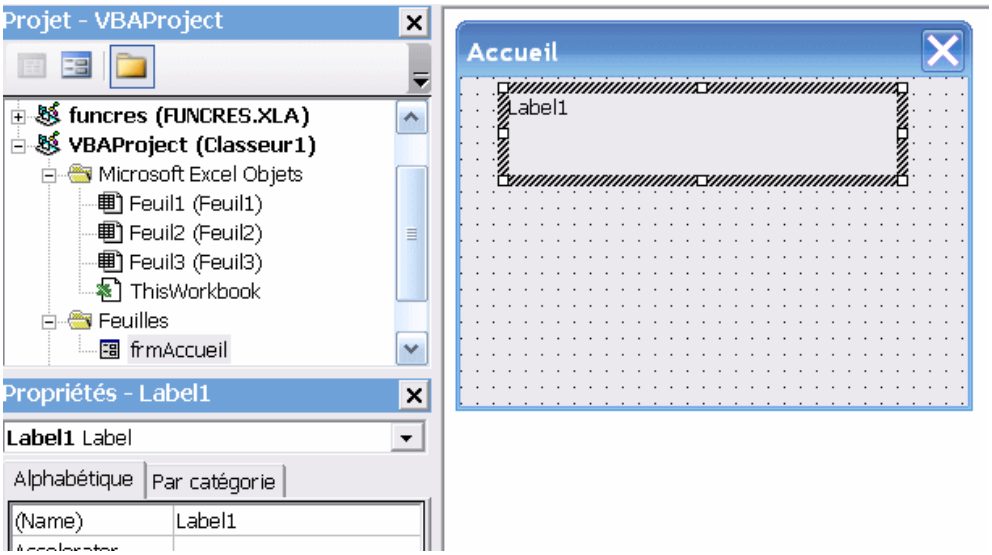


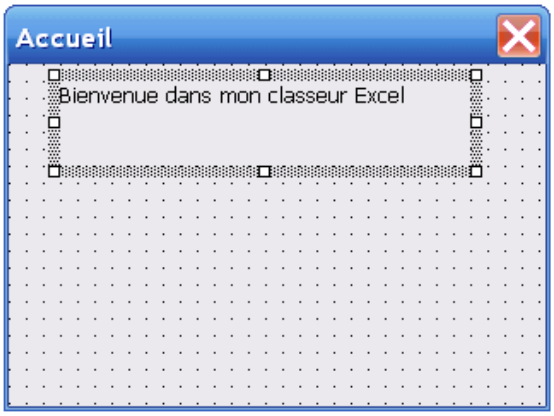
●Du menu **Insertion**, sélectionnez **UserForm**:

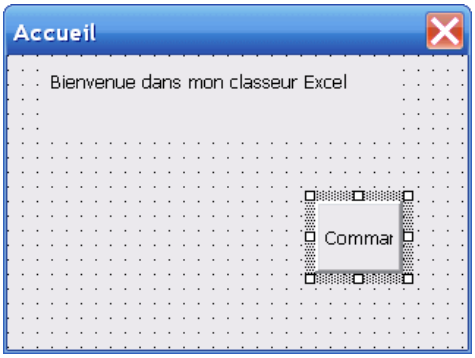

Microsoft Visual Basic - Classeur1 - [frmAccueil (UserForm)]
Fichier Edition Affichage Insertion Format Débogage Exécution Outils Compl
Projet - VBAProject
funcres (FUNCRES.XLA)
VBAProject (Classeur1)
Microsoft Excel Objets
Feuil1 (Feuil1)
Feuil2 (Feuil2)
Feuil3 (Feuil3)
ThisWorkbook
Feuilles
frmAccueil
UserForm1
Propriétés - frmAccueil
frmAccueil UserForm
Alphabétique
Par catégorie
(Name)
frmAccueil

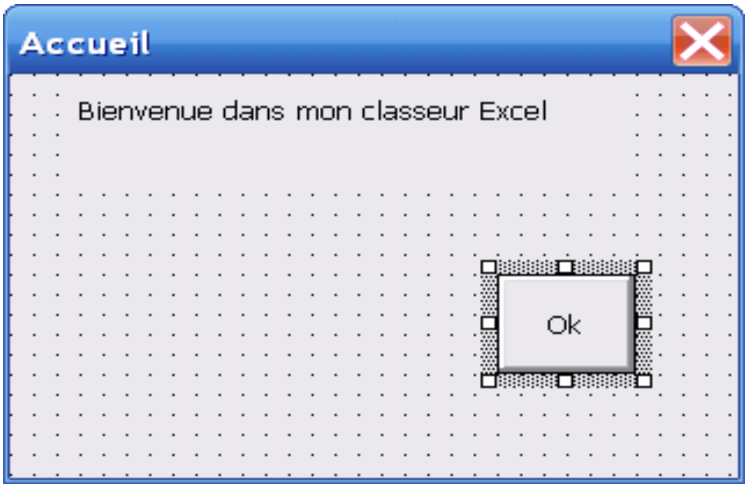

Microsoft Visual Basic - Classeur1 - [frmAccueil (User
Fichier Edition Affichage Insertion Format Débogage
Projet - VBAProject
funcres (FUNCRES.XLA)
VBAProject (Classeur1)
Microsoft Excel Objets
Feuil1 (Feuil1)
Feuil2 (Feuil2)
Feuil3 (Feuil3)
ThisWorkbook
Feuilles
frmAccueil
Accueil
Propriétés - frmAccueil
frmAccueil UserForm
Alphabétique
Par catégorie
(Name)
frmAccueil
BackColor
&H8000000F&
BorderColor
&H80000012&
BorderStyle
0 - fmBorderStyleN
Caption
Accueil

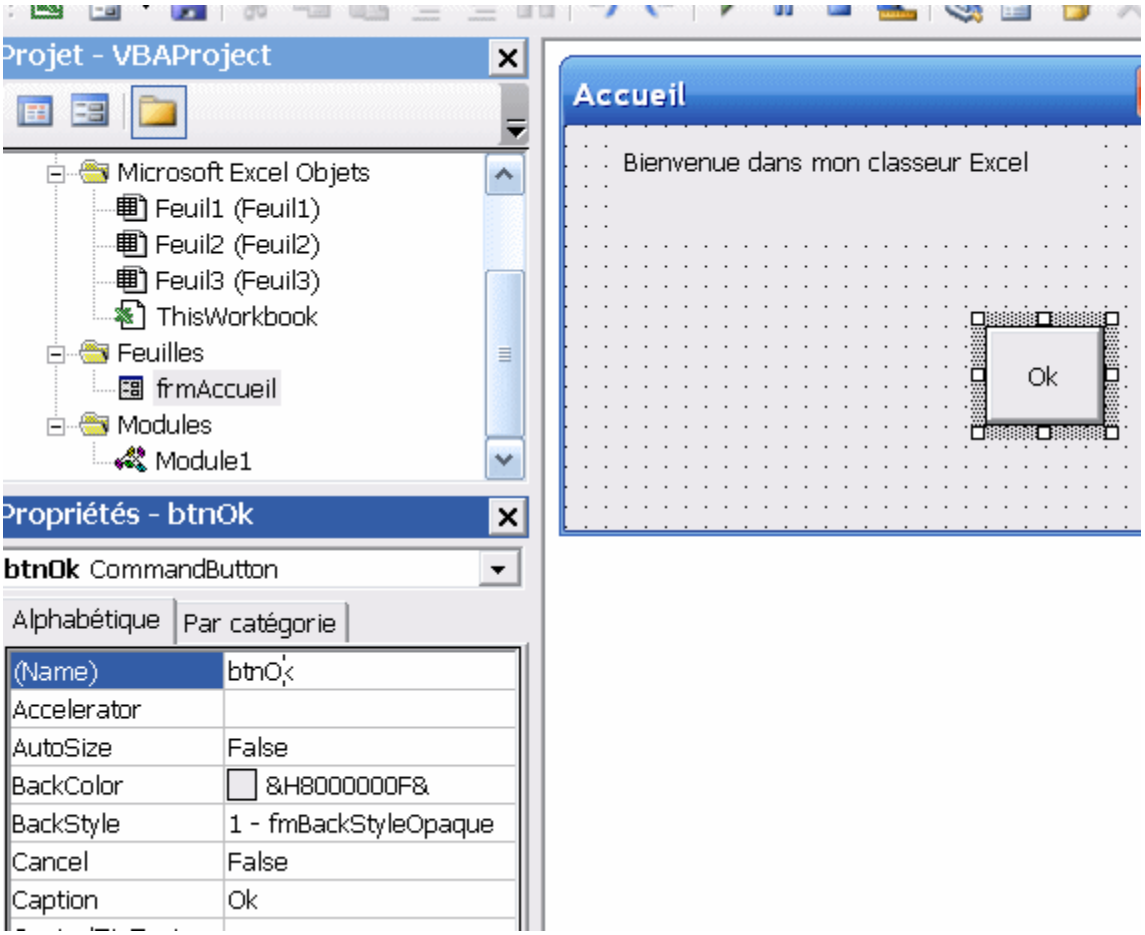

Boîte à outils
Contrôles

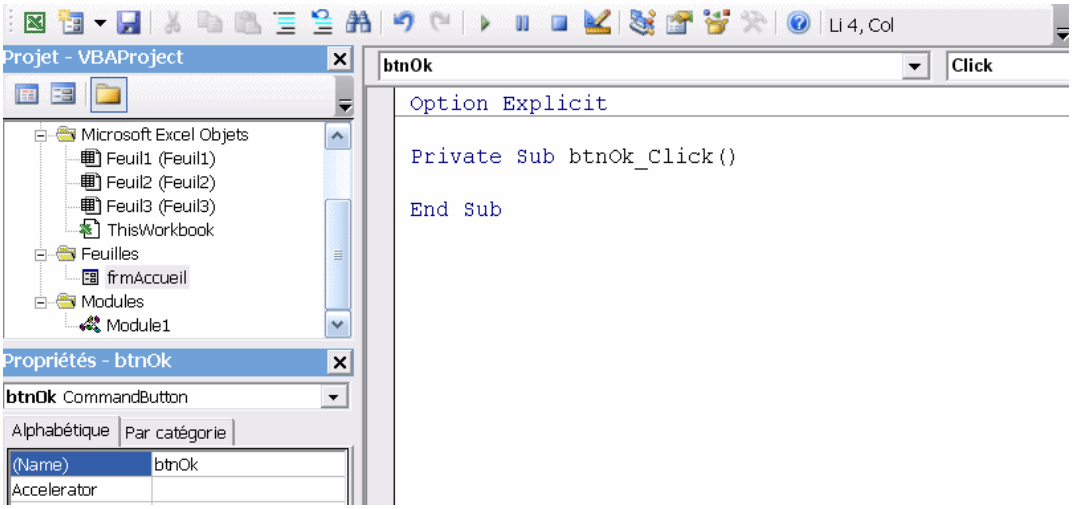

Projet - VBAProject
funcres (FUNCRES.XLA)
VBAProject (Classeur1)
Microsoft Excel Objets
Feuil1 (Feuil1)
Feuil2 (Feuil2)
Feuil3 (Feuil3)
ThisWorkbook
Feuilles
frmAccueil
Propriétés - Label1
Label1 Label
Alphabétique
Par catégorie
(Name)
Label1
Accueil
Label1


Accueil
Bienvenue dans mon classeur Excel

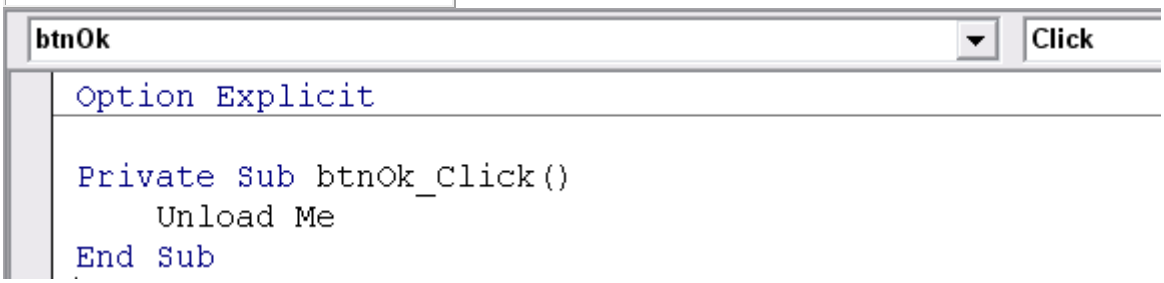

Accueil
Bienvenue dans mon classeur Excel
Commar


Accueil
Bienvenue dans mon classeur Excel
Ok

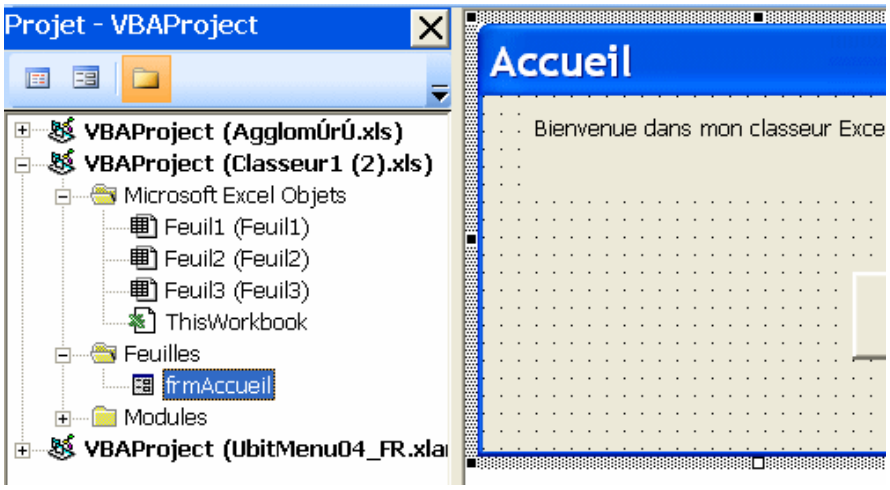

Projet - VBAProject
Microsoft Excel Objets
Feuil1 (Feuil1)
Feuil2 (Feuil2)
Feuil3 (Feuil3)
ThisWorkbook
Feuilles
frmAccueil
Modules
Module1
Propriétés - btnOk
btnOk CommandButton
Alphabétique
Par catégorie
(Name) btnOk
Accelerator
AutoSize False
BackColor &H8000000F&
BackStyle 1 - fmBackStyleOpaque
Cancel False
Caption Ok
Accueil
Bienvenue dans mon classeur Excel
Ok




```
Private Sub btnOk_Click()
    Unload Me
End Sub
```



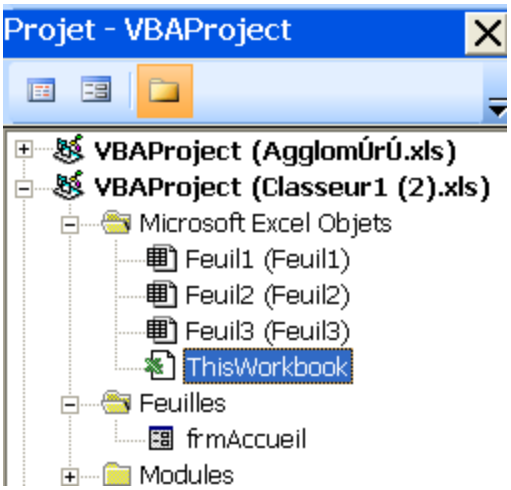
Il faut maintenant testez le formulaire.



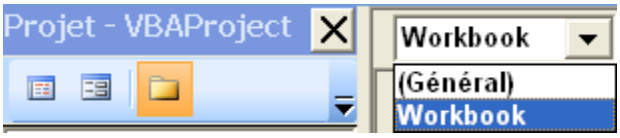
- Double-cliquez sur **frmAccueil** pour vous assurer que le formulaire est affiché en mode création:
- Appuyez sur la touche **F5** pour exécuter la boîte de dialogue.
- Appuyez sur le bouton 

pour la fermer.

Associez l'ouverture du formulaire à l'ouverture du classeur Excel:

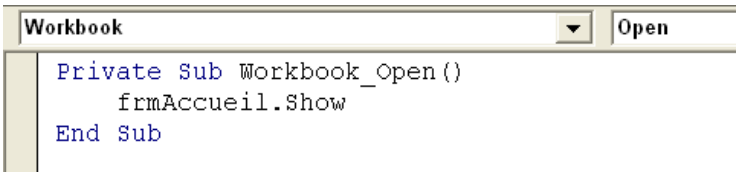


Dans l'explorateur de projet, double-cliquez sur **ThisWorkbook**:



Dans la liste déroulante (général), choisir **Workbook**:

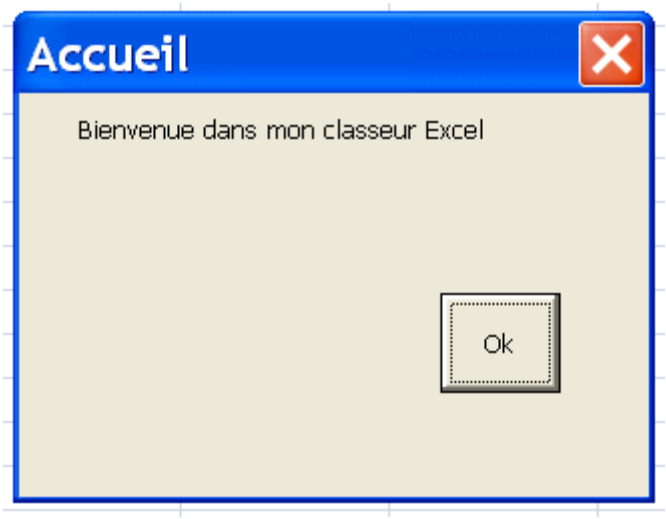
La seconde liste déroulante affiche **Open**.



Dans la zone de texte, inscrire **frmAccueil.Show** entre les lignes **Private Sub** ... et **End Sub**:

Fermez Visual Basic, sauvegardez le classeur Excel et fermez-le.
Ouvrez à nouveau le classeur. Il est probable que vous devrez activer les macros.

L'écran d'accueil s'affichera alors automatiquement.



Vous pouvez le fermer en appuyant sur **OK**.

Quelques commentaires sur le programme:

♦Il illustre le dernier des 3 types de programme qu'on peut réaliser avec VBA: une boîte de dialogue (ou formulaire).
♦Il contient des objets de formulaire

- Un formulaire
- Un libellé
- Un bouton

♦Il contient deux procédures événementielles:

- Sub Workbook_Open() est exécutée lors de l'ouverture du classeur.
- Sub btnOk_Click() est exécutée lorsque l'utilisateur appuie sur le bouton **OK**.

♦Il contient plusieurs catégories d'instructions:

- Une méthodes: Show de l'objet frmAccueil
- Une instruction VBA : Unload

♦Il contient aussi plusieurs maladresses de programmation, la plus grande étant qu'il ne s'efface pas de lui-même après quelques instants. Sans compter qu'il n'est pas très joli.

VBA pour Excel - Concepts de programmation

Caractéristiques d'un bon programme

Un programme informatique est un peu comme une recette de cuisine. Il requiert des ingrédients (données, variables, objets), contient une série de manipulations à faire avec ces ingrédients (instructions VBA) en utilisant des outils plus ou moins performants (VBE, Excel, Windows) et donne un résultat quelquefois plus intéressant que les ingrédients pris séparément.

Il y a de bonnes recettes, et de moins bonnes recettes. Il en est de même pour les programmes.

Un bon programme:

- Fait **toujours** ce qui est prévu.
- Ne fait **jamais** ce qui n'est pas prévu.
- Est facile à **utiliser**.
- Est facile (peu coûteux) à **modifier**.

Très peu de programmes disponibles sur le marché actuel répondent à ces critères.
Lorsque vous aurez terminé l'écriture d'un programme, demandez-vous s'il répond aux quatre critères de façon satisfaisante pour l'usage prévu.

Une suggestion dès le départ: gardez vos programmes simples.

Types de programme

Du point de vue de l'utilisation (et donc de la programmation), un programme **VBA** Excel peut être:

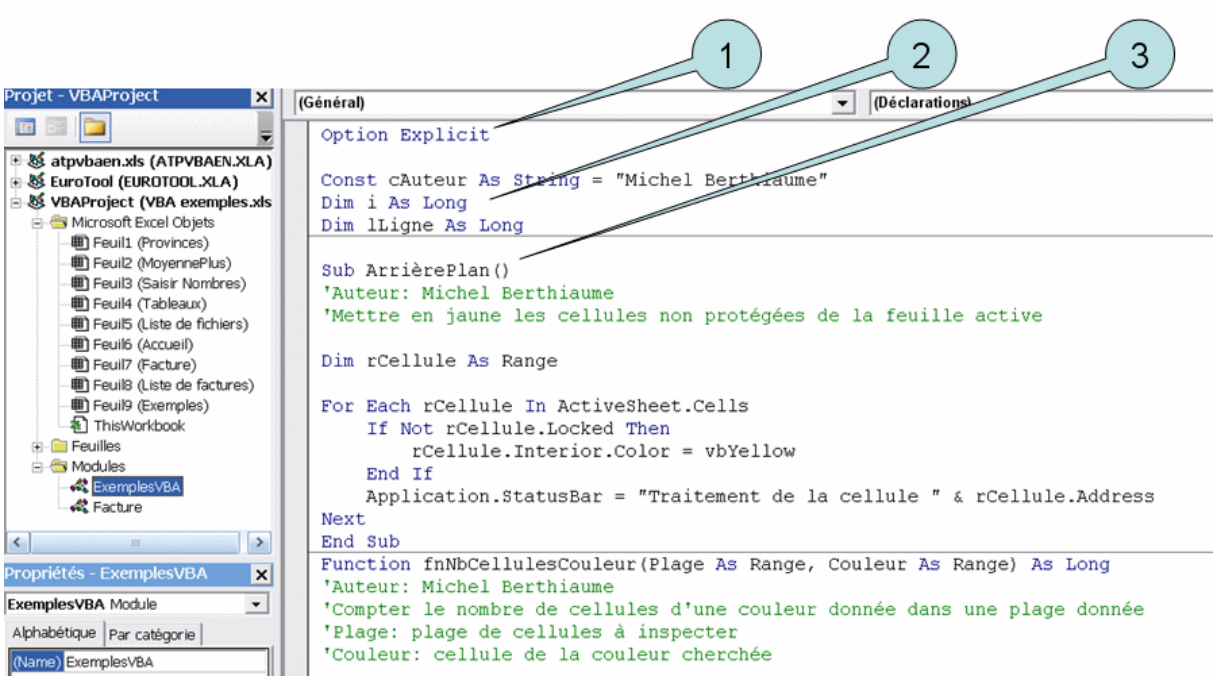
♦L'automatisation d'une série d'opérations qu'on peut faire manuellement dans Excel: la procédure **SUB**.
♦Une fonction Excel que les programmeurs Microsoft n'ont pas prévue: la procédure **Function** (fonction personnalisée).
♦Une boîte de dialogue qui permet des échanges avec l'utilisateur autrement que par les moyens prévus par les programmeurs Microsoft: le formulaire.
♦La création d'une classe.

On trouve un exemple de chacun (sauf la création de classe) dans la page Introduction du présent tutoriel.

Composition d'un module VBA

Une module VBA se compose des trois éléments suivants:

1. L'instruction **Option Explicit**, qui force la déclaration explicite des variables.
2. Des déclarations de variables de niveau module.
3. Une ou des procédures (programmes):



Une procédure doit toujours être encadrée par les instructions **Sub...End Sub** ou **Function ... End Function**.

[**Private** | **Public**] [**Static**] **Sub** *nom* [*(liste de paramètres)*]
[instructions]
[**Exit Sub**]
[instructions]
End Sub

Où:

♦**Public** Indique que la procédure **Sub** est accessible à toutes les autres procédures dans l'ensemble des modules. Valeur par défaut.
♦**Private** Indique que la procédure **Sub** n'est accessible qu'aux autres procédures du module dans lequel elle a été déclarée.
♦**Static** Indique que les valeurs des variables locales de la procédure Sub sont conservées entre les appels.En l'absence de **Static**, les variables sont réinitialisées à chaque exécution de la procédure. **Static** est très rarement nécessaire.
♦*nom* Nom de la procédure Sub. Doit especter les règles des noms de variables.
♦*Liste de paramètres* Liste de variables représentant des paramètres qui sont passés à la procédure Sub lorsqu'elle est appelée. Les variables multiples sont séparées par des virgules. Chaque paramètre doit être déclaré dans la forme suivante:
[**Optional**] [**ByVal** | **ByRef**] [ParamArray] *nom*[( )] [As **type**] [=*défaut*]
où:

♦**Optional** indique que ce paramètre n'est pas obligatoire. S'il est utilisé, tous les paramètres suivants doivent l'être aussi.
♦**ByRef** indique que si la valeur du paramètre est modifiée dans la procédure, elle le sera aussi dans la procédure appelante.

ATTENTION, en VBA **ByrRef** est la valeur par défaut, au contraire de la plupart des langages de programmation.

♦**ByVal** indique que si la valeur du paramètre est modifiée dans la procédure, cela n'affecte pas sa valeur dans la procédure appelante.
♦**ParamArray** Indique que le paramètre suivant est un tableau. **ParamArray** ne peut précéder que le dernier paramètre de la liste.
♦*Nom* nom du paramètre, doit respecter les règles des noms de variables.
♦**type** type du paramètre.
♦*défaut* constante initialisant un paramètre optionnel (**Optional**)

♦*instructions* Tout groupe d'instructions à exécuter dans la procédure **Sub**.
♦**Exit Sub** Instruction VBA permettant de terminer l'exécution de la procédure avant la fin.
♦**End Sub** Instruction délimitant la fin de la procédure **Sub**.

[**Private** | **Public**] [**Static**] **Function** *nom* [*(liste de paramètres)*] [**AS** *type*]
[instructions]
[*nom= expression*]
[**Exit Function**]
[instructions]
[*nom= expression*]
End Function

Où:

♦**Public** Indique que la procédure **Function** est accessible à toutes les autres procédures dans l'ensemble des modules.
♦**Private** Indique que la procédure **Function** n'est accessible qu'à d'autres procédures du module dans lequel elle a été déclarée.
♦**Static** Indique que les valeurs des variables locales de la procédure Sub sont conservées entre les appels.En l'absence de **Static**, les variables sont réinitialisées à chaque exécution de la procédure. **Static** est très rarement nécessaire.
♦*nom* Nom de la procédure Function. Respecte les règles des noms de variables.
♦*Liste de paramètres* Liste de variables représentant des paramètres qui sont passés à la procédure Sub lorsqu'elle est appelée. Les variables multiples sont séparées par des virgules. Chaque paramètre doit être déclaré dans la forme suivante:
[**Optional**] [**ByVal** | **ByRef**] [ParamArray] *nom*[( )] [As **type**] [=*défaut*]

où:

- **Optional** indique que ce paramètre n'est pas obligatoire. S'il est utilisé, tous les paramètres suivants doivent l'être aussi.
- **ByRef** indique que si la valeur du paramètre est modifiée dans la procédure, elle le sera aussi dans la procédure appelante. **ATTENTION,** en VBA **ByRef** est la valeur par défaut, au contraire de la plupart des langages de programmation.
- **ByVal** indique que si la valeur du paramètre est modifiée dans la procédure, cela n'affecte pas sa valeur dans la procédure appelante.
- **ParamArray** Indique que le paramètre suivant est un tableau. **ParamArray** ne peut précéder que le dernier paramètre de la liste.
- *Nom* nom du paramètre, respectant les règles des noms de variables.
- **type** type du paramètre.
- *défaut* constante initialisant un paramètre optionnel (**Optional**)

♦*Type* Type de la valeur retournée par la procédure Function.
♦*instructions* Tout groupe d'instructions à exécuter dans la procédure **Function**

♦*expression* Valeur retournée par la procédure **Function**.
♦**Exit Function** Instruction VBA permettant de terminer l'exécution de la procédure avant la fin.
♦**End Function** Instruction délimitant la fin de la procédure **Function**.

Sub ou Function?

Une procédure **Sub** sans paramètre obligatoire:

♦Ne retourne pas de valeur.
♦Peut modifier le contenu de la feuille Excel.
♦Peut être associée à un objet (bouton, graphique) dans la feuille Excel et exécutée lorsque l'objet est cliqué.

Une procédure **Sub** avec paramètre obligatoire:

♦Ne retourne pas de valeur.
♦Peut être exécutée lorsqu'un événement Excel se produit (ouverture, fermeture, modification de cellule...).

Une procédure **Function**:

♦Retourne une valeur.
♦Ne doit pas modifier la feuille de travail.
♦Ne doit pas afficher de boîte de dialogue.
♦Ne peut être utilisée que dans une formule Excel ou une procédure VBA.

Composantes d'un programme

Un programme VBA comporte un nom, de la documentation, des objets, des instructions et une gestion des erreurs:

♦Un nom: Assignez toujours un nom qui décrit la nature de votre programme. Ça le rend plus facile à **modifier.**
♦De la documentation: au minimum, indiquez dès la 2e ligne le nom de l'auteur et dès la 3e ligne le but du programme. N'hésitez pas à parsemer votre programme de commentaires qui vous permettront de mieux comprendre ce qu'il fait (et comment il le fait) quand vous voudrez **modifier** le programme dans un ou dix ans.
♦Des objets: une recette contient des ingrédients, un programme contient des objets. Ces objets peuvent être:

- Des cellules Excel (leur contenu. leur couleur, leur format, ...).
- D'autres objets apparaissant à l'écran (graphique, feuille, onglet, imprimante, ...).
- Des valeurs variables (compteurs, accumulateurs, ...).
- Des valeurs constantes (Pi, code de couleur, ...).
- Une boîte de dialogue contient des contrôles, nom qu'on donne aux objets de formulaire qui apparaissent à l'écran (boutons, zones de texte, cases, ...).

♦Des instructions:

- Assignation, conversion.
- Action.

- Contrôle.

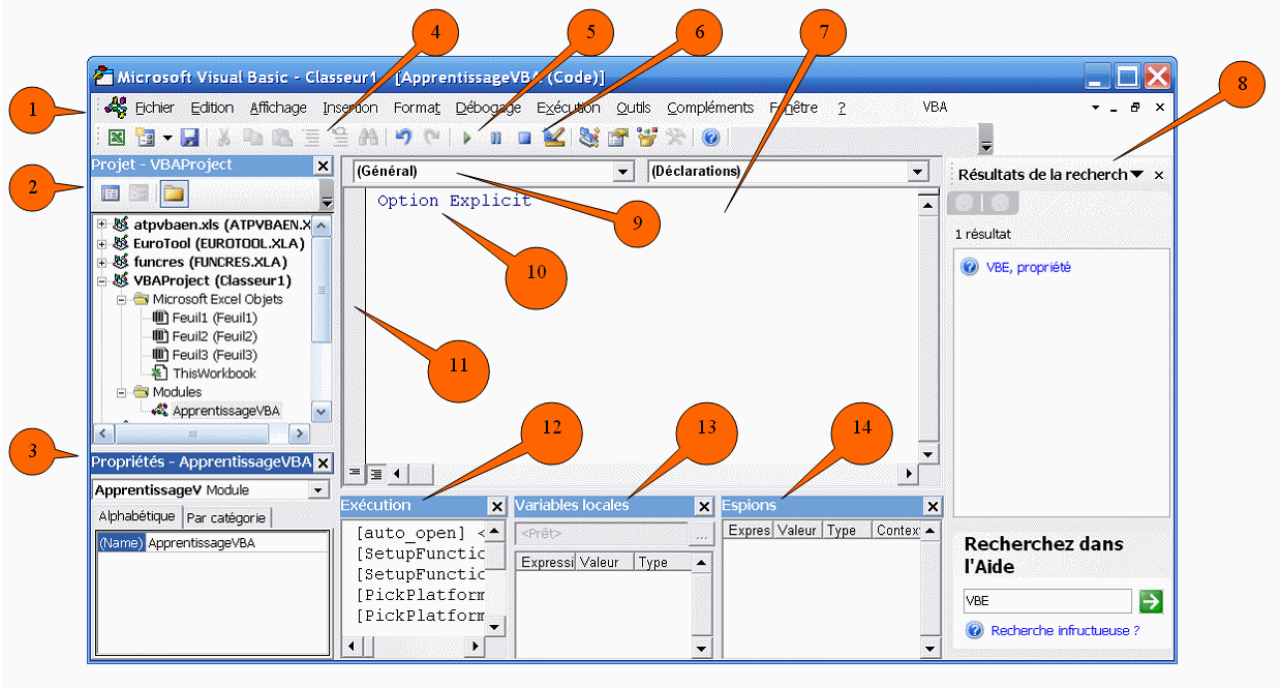
♦Une gestion des erreurs: des instructions à exécuter lors que l'inattendu se produit.

Pour développer un programme, on utilise un environnement de développement. Cet environnement est le même pour tous les programmes de la suite Office: VBE (Visual Basic Editor).

Le code d'un programme VBA est enregistré dans un classeur, dans une feuille, dans un module ou dans un formulaire. Le tout est enregistré à l'intérieur du document Office. En Office 2007, les noms des documents contenant un programme VBA ont une extension spécifique (.xlsm au lieu de .xlsx).

VBA pour Excel - VBE: l'éditeur VBA

Vue d'ensemble



Le tableau ci-dessous explique les éléments qu'on retrouve à l'écran. Utilisez le menu **Affichage** pour afficher les éléments absents de votre affichage.

1) Menus	Les choix les plus fréquemment utilisés sont dans les menus Insertion et Débogage.
2) Explorateur de projets	Sert à choisir ce qui est affiché dans la fenêtre de code (7) et dans l'explorateur d'objets (3).
3) Explorateur d'objets	Permet de consulter et modifier les propriétés de l'objet sélectionné dans l'explorateur de projets (2).
4) Boutons additionnels	Il est recommandé d'ajouter ces boutons à la barre d'accès rapide.
5) Bouton Démarrer/Continuer	Sert à démarrer l'exécution continue du code sélectionné dans la fenêtre de code (7).

6) Bouton Arrêter	Sert à arrêter l'exécution du code VBA en cours.
7) Fenêtre Code OU Fenêtre Formulaire	Affiche le code VBA **OU** les objets de formulaire de l'objet sélectionné dans l'explorateur de projets (2).
8) Fenêtre de recherche dans l'aide	Affiche les résultats de la recherche dans l'aide.
9) Nom du module affiché	Liste déroulante permettant de passer d'un objet du projet à un autre sans passer par l'explorateur de projets (2).
10) Option Explicit	Instruction que l'on doit retrouver au début de chaque module et qui force la déclaration explicite des variables.
11) Marge de la fenêtre Code	Contient des repères visuels lors du développement d'un programme VBA.
12) Fenêtre Exécution	
13) Fenêtre Variables locales	
14) Fenêtre Espions	

Modifications essentielles à l'interface

Les boutons Commenter bloc  et Ne pas commenter bloc  vous seront utiles pour rendre inopérantes/opérantes des instructions erronées ou suspectes lors de tests et débogages.
Pour les ajouter à la barre d'accès rapide :

- Enfoncez le bouton de droite de la souris sur une zone inutilisée de la barre d'accès rapide.
- Sélectionnez **Personnaliser...**
- Ouvrez l'onglet **Commandes.**
- Sélectionnez la Catégorie **Édition.**
- Dans la liste déroulante, choisissez **Commenter bloc**.
- Faites glisser l'icône à l'endroit choisi de la barre d'accès rapide.
- Répétez pour le bouton **Ne pas commenter bloc**.

L'instruction **Option Explicit** peut et doit apparaître au début de chaque module VBA.
Pour l'automatiser:

- Dans le menu **Outils**, choisissez **Options**.
- Dans l'onglet **Éditeur**, assurez-vous que la case à cochée **Déclaration des variables obligatoire** est cochée.

L'explorateur de projets (sauvegardes)

L'explorateur de projets permet de naviguer dans les différents endroits pouvant contenir du code VBA Excel,
c'est à dire les différents emplacements où il vous est possible de sauvegarder le code que vous allez écrire.

Vos choix:

♦Un classeur Excel: le code VBA est sauvegardé avec le classeur. Si vous copiez le classeur, il est copié avec. Si vous supprimez le classeur, il est supprimé.

♦Le classeur de macros personnelles: le code VBA est sauvegardé dans le classeur personal.xlsb de l'ordinateur, **ET NON DANS LE CLASSEUR ACTIF**. Comme personal.xlsb est automatiquement chargé lors du démarrage d'Excel sur un ordinateur, les programmes VBA sauvegardés dans le classeur personal.xlsb de cet ordinateur sont disponibles dans tous les classeurs ouverts sur ce même ordinateur. Ce choix est utile aux programmeurs expérimentés et aux administrateurs de parcs d'ordinateurs. **Il est déconseillé aux programmeurs amateurs.**

Dans un classeur Excel, vous devez choisir un des endroits suivants:

♦Une feuille de ce classeur. Cet emplacement déconseillé.

♦Le classeur lui-même (**ThisWorkbook**). Il est utilisé pour enregistrer les programmes déclenchés par des événements de classeur (ouverture, fermeture, modification de cellule...).

♦Un module. Il est utilisé pour enregistrer les programmes utilisés dans le classeur ou par les autres programmes du classeur.

♦Un formulaire. Il est utilisé pour enregistrer les programmes déclenchés par les événements du formulaire.

Tous ces emplacements peuvent contenir du code VBA Excel.

En pratique, vous utiliserez le plus souvent les endroits suivants:

♦Un module dans un classeur Excel.
♦Un formulaire dans un classeur Excel.

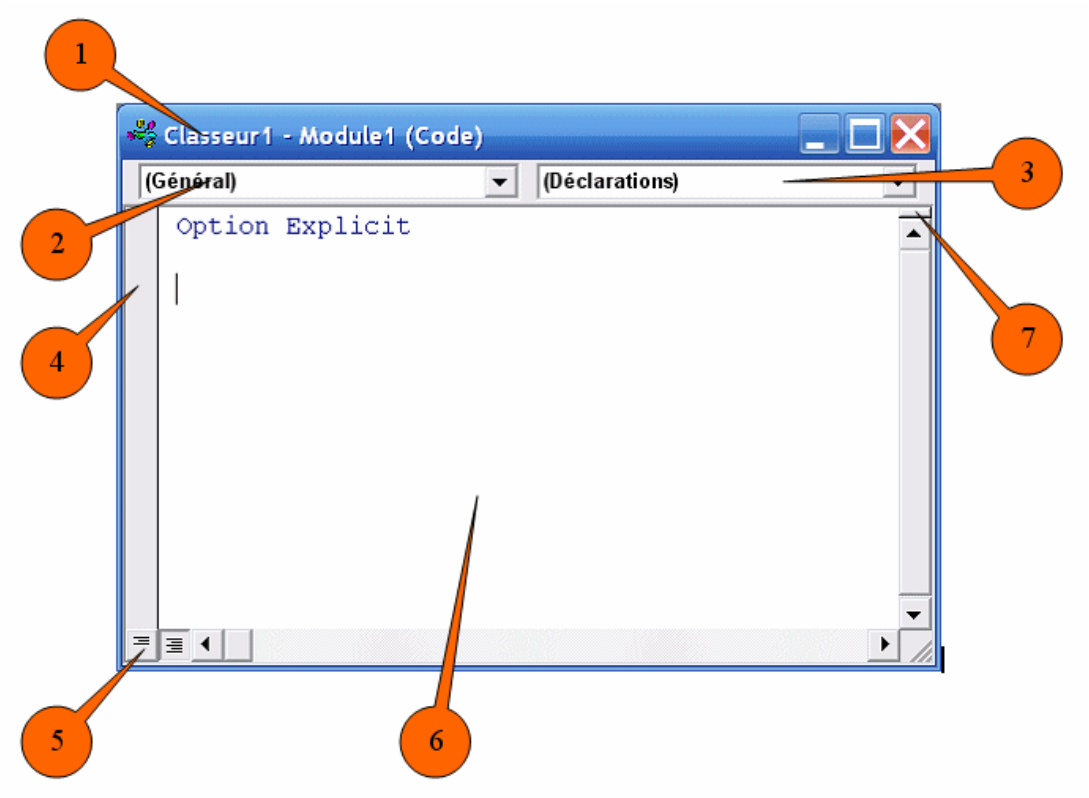
Explorateur d'objets (fenêtre Propriétés)

L'explorateur d'objets sert principalement:

♦À renommer les modules et formulaires.
♦À modifier les propriétés des contrôles (objets) de formulaires

Fenêtre Code ou fenêtre Formulaire

C'est dans cette fenêtre qu'on crée/édite le code VBA.



1) Entête

Affiche le nom de l'emplacement (classeur, feuille de classeur, module, formulaire) contenant le code affiché. L'emplacement est choisi dans l'explorateur de projets.

2) Zone objet

Objets disponibles dans le contexte du conteneur choisi dans l'explorateur de projets.

3) Zone procédures/ événements

Liste de toutes les procédures du module (en gras) et des procédures événementielles (en pâle) possibles de l'objet choisi dans la zone objet (2).

4) Marge de la fenêtre code

Contient des repères visuels lors du développement d'un programme VBA: les points d'arrêt.

5) Affichage Procédure/module

Boutons permettant de basculer entre l'affichage d'une seule procédure et l'affichage de toutes les procédures du module.

6) Éditeur de code

Éditeur de texte permettant de créer/éditer des procédures SUB ou des procédures Function.

7) Barre de fractionnement

Permet de diviser la fenêtre de code en deux parties horizontales pour afficher simultanément deux parties d'un même module.

La fenêtre code offre les fonctionnalités d'éditeur habituelles pour créer/éditer du code VBA. Il y existe aussi quelques fonctionnalités additionnelles:

♦Les touches **Ctrl+espace** offrent de compléter le mot où se trouve le curseur dans la fenêtre de code.

♦Les mots réservés VBA sont bleus.
♦Les commentaires sont verts.
♦Les lignes erronées sont rouges.
♦La touche **F1** démarre l'aide contextuelle, dont le principal défaut est l'absence d'outil de recherche. Voir la fenêtre recherche.

D'autres raccourcis sont disponibles.

Fenêtre Exécution

Cette fenêtre permet d'exécuter une ligne de code VBA en dehors d'un module.

Par exemple, taper ? fnMoyenneAvecCellulesNulles(Range("A1..B5") exécute la fonction personnalisée fnMoyenneAvecCellulesNulles et en affiche le résultat.

? Range("A1") affichera la valeur de la cellule A1. C'est modérément utile pour tester des instructions.

Fenêtre variables locales

Cette fenêtre affiche la liste des variables locales du module en cours d'exécution (mode débogage) et permet d'en connaître la valeur et le type.

Fenêtre Espions

Cette fenêtre affiche la valeur de variables choisies, en mode débogage. À la différence de la Fenêtre variables locales, il est possible de s'y faire

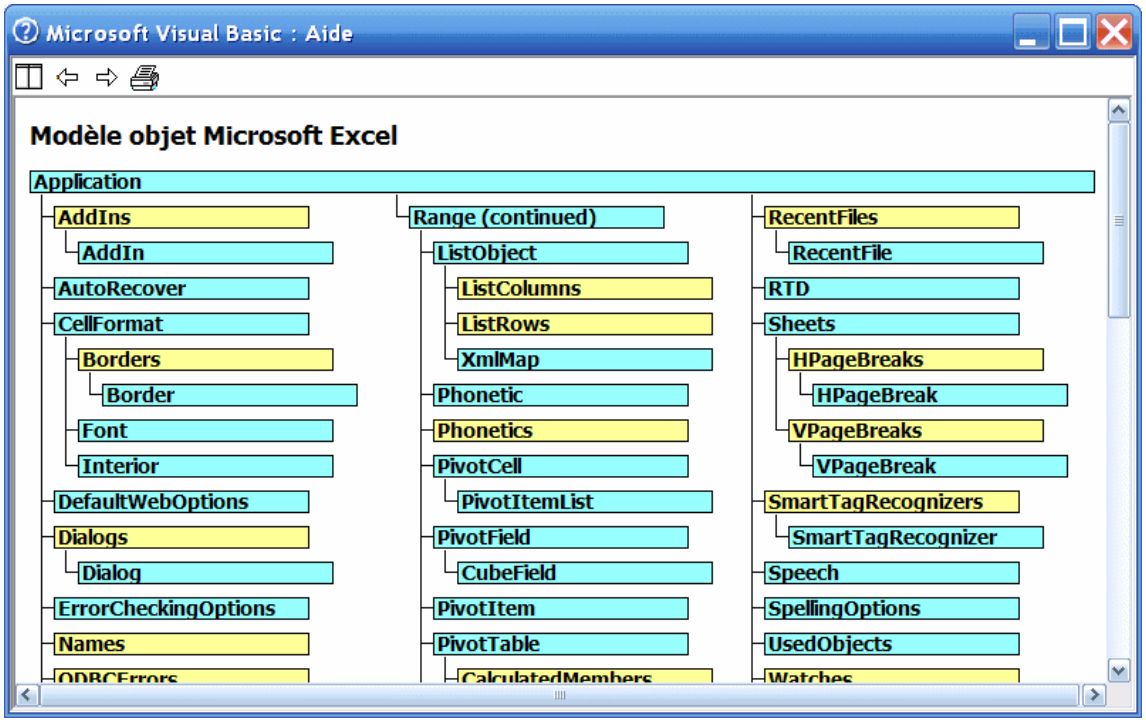
afficher la valeur de n'importe quelle variable, locale ou non.

Fenêtre Recherche (Aide)

Cette fenêtre donne accès à l'aide VBA locale et en ligne, et affiche le résultat d'une recherche.
Pour afficher la fenêtre Recherche, faites une recherche dans la zone de texte déroulante **Taper une question** de la barre de menus (en haut à droite de la fenêtre VBA).

Vous pouvez également démarrer l'aide contextuelle à partir de la fenêtre de code, en utilisant la touche **F1**, dont le principal défaut est l'absence d'outil de recherche.

En faisant une recherche sur le mot clé **Objet Excel**, vous accéderez à une description graphique des objets et des collections Excel que votre programme VBA peut manipuler:



Raccourcis utiles en VBE

♦**F7** affiche la fenêtre code.
♦**F2** affiche l'explorateur d'objets.
♦**Ctrl+bas** procédure suivante.
♦**Ctrl+PgDn** écran suivant.
♦Autres raccourcis habituels (**Ctrl-C**, **Tab**, etc...).
♦**F9** ajoute/supprime un point d'arrêt.
♦**Ctrl+Maj+F9** supprime tous les points d'arrêt.
♦**Ctrl+espace** active Compléter mot.
♦**F5** exécute la procédure active.
♦**F8** démarre le mode pas à pas.
♦**Ctrl+F8** exécute jusqu'au curseur.
♦**Ctrl+S** enregistre le code VBA dans le classeur (mais n'enregistre pas le classeur!).

Déboguer

Un programmeur passe la grande majorité de son temps à tester et à déboguer du code. Il est donc essentiel de maîtriser les techniques de débogage ci-dessous.

Pour une procédure **Sub** sans paramètre:

- Placez le curseur dans la procédure.
- Appuyez la touche **F8** pour exécuter une ligne de code.
- Examinez les résultats de cette exécution dans la fenêtre variables, locales, dans la fenêtre espions, en survolant une variable de la procédure avec la souris ou dans le classeur Excel.
- Appuyez la touche **F8** pour exécuter la ligne de code suivante.
- Répétez ...

Alternative:

- Placez le curseur dans la procédure, sur la ligne qui vous inquiète.
- Appuyez les touches **Ctrl+F8** pour exécuter la procédure jusqu'à cette ligne.
- Examinez les résultats de cette exécution dans la fenêtre variables locales et dans la fenêtre espions, en survolant une variable de la procédure avec la souris ou dans le classeur Excel.
- Appuyez la touche **F8** pour exécuter la ligne de code suivante.
- Répétez **F8** pout chaque ligne que vous voulez exécuter.

Alternative:

- Placez un point d'arrêt dans la marge à la hauteur de la ligne qui vous inquiète (**F9** ou Débogage/point d'arrêt).
- Démarrez l'exécution (**F5**).
- Examinez les résultats de cette exécution jusqu'à la ligne contenant le point d'arrêt dans la fenêtre variables locales et dans la fenêtre espions, en survolant une variable de la procédure avec la souris ou dans le classeur Excel.
- Appuyez la touche **F8** pour exécuter la ligne de code suivante.
- Répétez **F8** pout chaque ligne que vous voulez exécuter.

Alternative:

- Placez l'instruction **Stop** dans une nouvelle ligne avant la ligne qui vous inquiète.
- Démarrez l'exécution (**F5**).
- Elle s'arrête à la line **Stop**.
- Appuyez la touche **F8** pour exécuter la ligne de code suivante.
- Examinez les résultats de cette exécution dans la fenêtre variables locales et dans la fenêtre espions, en survolant une variable de la procédure avec la souris ou dans le classeur Excel.
- Répétez la touche **F8** pout chaque ligne que vous voulez exécuter.

Alternative si l'exécution de la procédure semble s'éterniser:

- Placez le curseur dans la procédure.
- Démarrez l'exécution (**F5**).
- Stoppez l'exécution avec les touches **Ctrl+Pause** (Attn ou Break).
- Utilisez la touche **F8** pour exécuter les lignes de code une à une.
- Examinez les résultats de l'exécution de chaque ligne dans la fenêtre variables locales et dans la fenêtre espions, en survolant une variable de la procédure avec la souris ou dans le classeur Excel.
- Répétez **F8** pout chaque ligne que vous voulez exécuter.

Pour une procédure **Sub** avec paramètre ou pour une procédure **Function** (fonction):

- Placez un point d'arrêt près d'une ligne suspecte de la procédure ou la fonction.

OU

- Placez l'instruction **Stop** près d'une ligne suspecte de la procédure ou la fonction.

- Exécutez la procédure ou la fonction en l'appelant à partir de la fenêtre Exécution.