Micro-ordinateurs,
informations, idées, trucs et astuces

Utiliser Excel

Auteur : ***François CHAUSSON***
Date : ***8 février 2008***
Référence : ***utiliser Excel.doc***

Préambule

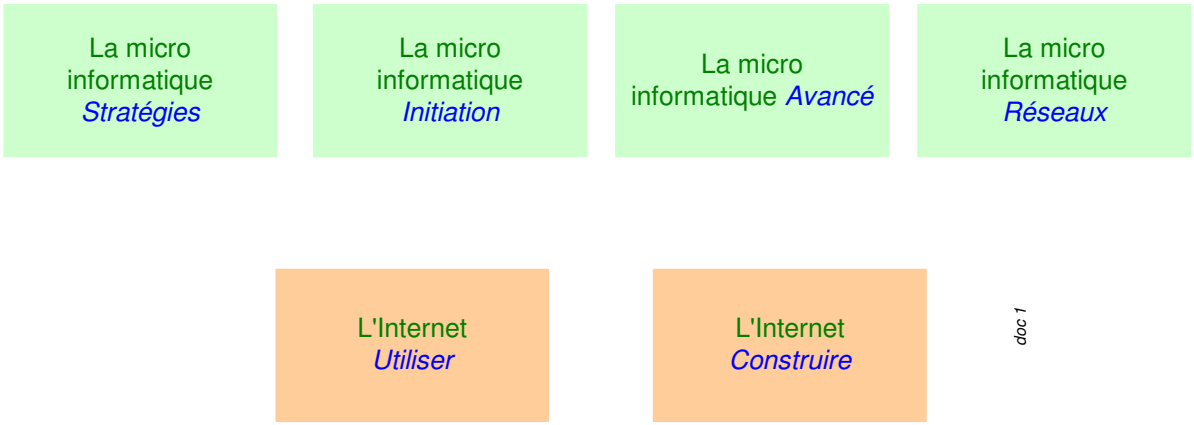
Voici quelques informations utiles réunies ici initialement pour un usage personnel en espérant qu'elles puissent aider d'autres utilisateurs de micro-informatique.

Ces informations sont présentées sans démarche pédagogique ; si un niveau de détail était nécessaire sur un sujet particulier, ne pas hésiter à me demander.

Ce document

Il fait partie de l'ensemble documentaire ***Micro-ordinateurs, informations, idées, trucs et astuces*** qui couvre ces sujets :

1. *La micro-informatique*, en 2 tomes
2. *L'Internet*, en 2 tomes



3. *Des Trucs HTML et Javascript*
4. *Des notices d'utilisation de divers logiciels*[1]

Tout commentaire à propos de ce document pourrait être adressé à :
pcinfosmicro@francois.chausson.name

Ce document est régulièrement mis à jour sur : http://fcfamille.free.fr/[2]

[1] ZoneAlarm, AVG, …
[2] Site à accès contrôlé

Table des matières

Excel

Voici quelques petits tuyaux utiles à l'utilisation de Excel.

Chargement automatique d'une feuille au démarrage

Pour provoquer le chargement automatique d'une feuille au démarrage de Excel :

- Dans l'explorateur, créer un raccourci sur la feuille Excel choisie
- Ouvrir le répertoire *C:\Program Files\Microsoft Office\OFFICE11\XLSTART*
- Ou dans *C:\Documents and Settings*\nom_utilisateur*Application Data\Microsoft\Excel*
- Faire *Copier/Coller* du raccourci dans ce répertoire

Macros

Source : http://www.01net.com/editorial/310812/programmation/automatiser-excel-grace-a-vba/

> Il est toujours agaçant, et même parfois pénible, de devoir effectuer des tâches répétitives, surtout quand celles-ci se résument à des suites d'actions sans aucun intérêt... C'est non seulement une perte de temps, mais aussi une source d'erreur, la répétition ayant tendance à diminuer l'attention. Et bien que l'informatique ait été en partie inventée pour traiter des tâches répétitives, nous devons encore trop souvent enchaîner les mêmes actions élémentaires au quotidien au sein de bon nombre de logiciels.

Heureusement, il existe une solution pour alléger cette corvée, du moins dans la suite Office de Microsoft. En effet, dans tous les programmes de cette suite bureautique, vous pouvez automatiser des opérations en créant ce que l'on appelle des macrocommandes (ou, plus simplement, des macros) : des suites d'instructions écrites avec Visual Basic pour Application (plus connu sous l'acronyme de VBA), un véritable langage de programmation. Intégré à tous les logiciels de la suite, VBA ne nécessite aucun ajout (il n'y a rien à télécharger ni à installer sur le PC). En revanche, comme tous les langages de programmation, il nécessite un apprentissage, ainsi que de la rigueur et beaucoup de patience ! Mais rassurez-vous, son fonctionnement et sa mise en oeuvre restent simples.

La présence d'un langage de macros dans Office n'est pas nouvelle. Mais les anciennes versions de ce langage, différentes pour chaque programme de la suite, ne permettaient qu'un enchaînement de tâches élémentaires. Commun à tous les logiciels d'Office 2000, XP ou 2003 à quelques nuances près, VBA change tout : on passe d'un système de macros à un véritable langage de programmation, avec lequel vous pouvez créer des logiciels simples ou complexes. Mais, l'habitude étant prise, on continue de donner le nom de macros aux programmes écrits en VBA. Il ne s'agit pas, pour autant, d'un langage universel car les macros écrites en VBA ne sont pas autonomes : elles ne fonctionnent qu'avec les logiciels de la suite Office. En d'autres termes, si vous pouvez librement copier vos macros d'un ordinateur sur un autre, elles ne fonctionneront sur un PC que si Office y est installé.

Des macros pour combler les lacunes de certains logiciels

Pourquoi écrire des macros ? D'abord, nous l'avons vu, pour enchaîner automatiquement, rapidement et sans risque d'erreur des commandes de logiciels Office. Mais aussi pour combler les lacunes de vos programmes. Ainsi, aucun menu d'Excel ne sait supprimer les lignes vides dans une feuille, ni renommer automatiquement tous les onglets d'un classeur... Autant de lacunes que des macros peuvent facilement combler. Enfin, pour les plus expérimentés, les macros permettent de créer de véritables logiciels, utilisables par des néophytes et dotés d'une interface semblable à celle de tous les programmes sous Windows, un aspect que nous n'aborderons toutefois pas dans le cadre de ce guide, faute de place.

Même si VBA reste plus simple que des langages pour développeurs tels que le C, le C++ ou bien encore l'Assembleur, il utilise des structures classiques de programmation, comme les boucles, les instructions conditionnelles, les variables... Autant de notions que nous

expliquerons, mais que vous parviendrez plus facilement à maîtriser si vous avez déjà une première expérience, même sommaire, d'un langage de programmation quelconque.

Notre but, dans ce guide, est de vous faire découvrir les bases de VBA. Après avoir passé en revue les étapes de la création et du lancement d'une macro, nous vous proposons une vingtaine d'exemples pratiques prêts à l'emploi, conçus pour Excel XP ou 2003 et triés par difficulté croissante. Pourquoi seulement pour Excel ? Parce que c'est le tableur qui se prête le mieux à une automatisation des tâches. De plus, si les bases de VBA restent les mêmes pour tous les programmes de la suite Office, il existe toutefois, d'un logiciel à l'autre, de sensibles différences dans les moyens à utiliser pour concevoir et exploiter les macros.

Bien entendu, vous pourrez adapter les macros que nous vous donnons en exemple à votre utilisation personnelle, mais aussi les optimiser. Nous les avons en effet conçues bien plus dans un but pédagogique que dans un souci de rapidité ou d'efficacité. Pour vous éviter de saisir ces listings, nous vous proposons de télécharger ceux-ci sur notre site. Le fichier joint LisezMoi.doc contient toutes les explications nécessaires à leur utilisation.
Comme vous le verrez, bien qu'il soit puissant et polyvalent, VBA demeure un langage relativement accessible. En apprendre les bases aujourd'hui peut constituer un investissement extrêmement rentable, tant les macros vous permettront de gagner du temps jour après jour.

Avec l'Enregistreur de macros

Faites une répétition

Comme vous allez le voir, l'Enregistreur mémorise toutes les actions que vous effectuez, sans la moindre intelligence. Il notera donc aussi fidèlement les « bonnes » manipulations... que vos erreurs. Vous devez donc, avant de démarrer, connaître parfaitement la suite des commandes que vous allez mémoriser.

Nous vous conseillons même de la « répéter » plusieurs fois sur un modèle pour en être sûr. Dans notre exemple, il faudra cliquer sur le bouton **G** de la barre d'outils **Mise en forme** pour mettre les cellules sélectionnées en gras, puis cliquer sur le bouton **I** de la même barre pour activer l'attribut Italiques et enfin cliquer sur le bouton **Centrer** de la barre d'outils **Mise en forme.**

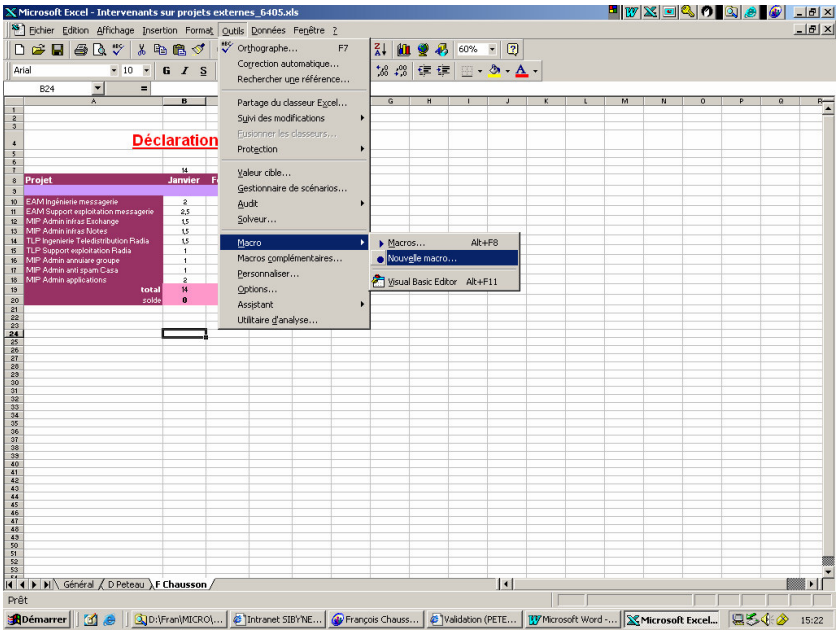
Préparez le terrain

La macro fait exactement ce que vous lui demandez, rien de plus, rien de moins. Ainsi, si elle doit traiter une zone sélectionnée, il faut, avant de démarrer l'enregistrement, ouvrir une feuille et sélectionner un champ de cellules. De façon générale, vous devrez faire « à la main » tout ce que la macro ne fera pas.

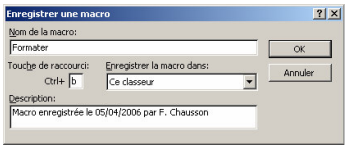
Dans notre exemple, lancez Excel, ouvrez un classeur quelconque, activez une feuille et sélectionnez un groupe de cellules non vides.

Lancez l'enregistrement

Déroulez le menu *Outils, Macros, Nouvelle macro*.



Dans le champ *Nom de la macro*, écrivez un libellé court, par exemple Formater.



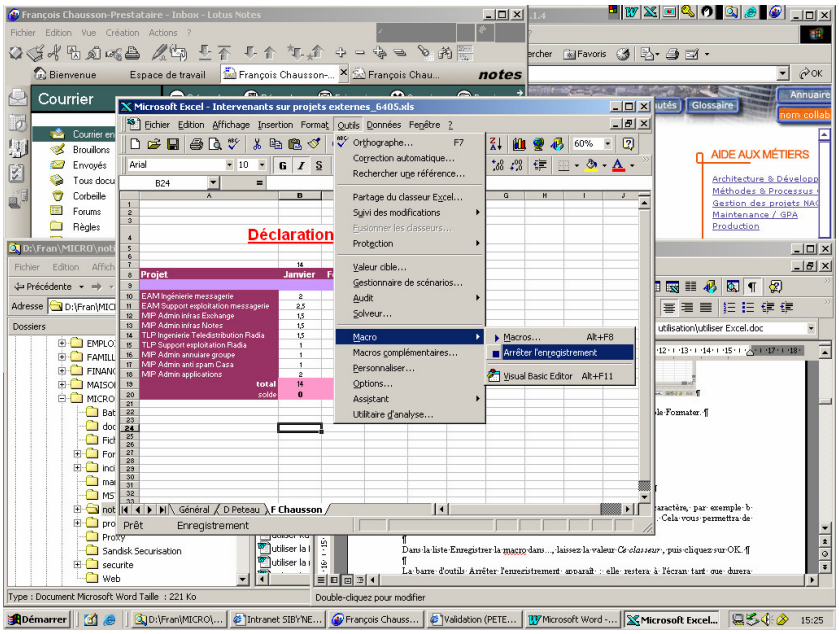
Dans le champ Touche de raccourci, tapez éventuellement un caractère, par exemple b (attention : les majuscules et les minuscules ne sont pas identiques). Cela vous permettra de lancer plus facilement la macro par la suite.

Dans la liste Enregistrer la macro dans..., laissez la valeur *Ce classeur* , puis cliquez sur OK.

La barre d'outils *Arrêter l'enregistrement* apparaît : elle restera à l'écran tant que durera l'enregistrement de la macro. Attention : tout ce que vous faites maintenant dans Excel est enregistré dans la macro.

Faites une par une les actions à enregistrer.

Dans la barre d'outils **Mise en forme** , cliquez successivement sur les boutons **G** (gras) et **I** (italique). Puis cliquez sur le bouton Centrer de la barre d'outils **Mise en forme :**



Votre macro est terminée.

Arrêtez l'enregistrement.

Cliquez sur bouton **Arrêter l'enregistrement** dans la petite barre d'outils flottante. Cette barre disparaît.

Exécutez la macro.

Sélectionnez maintenant n'importe quelle autre plage de cellules dans la même feuille et tapez **Ctrl + b** (le raccourci que vous aviez choisi lors de l'enregistrement de la macro). Les cellules sélectionnées sont mises en caractères gras, italiques puis centrées.

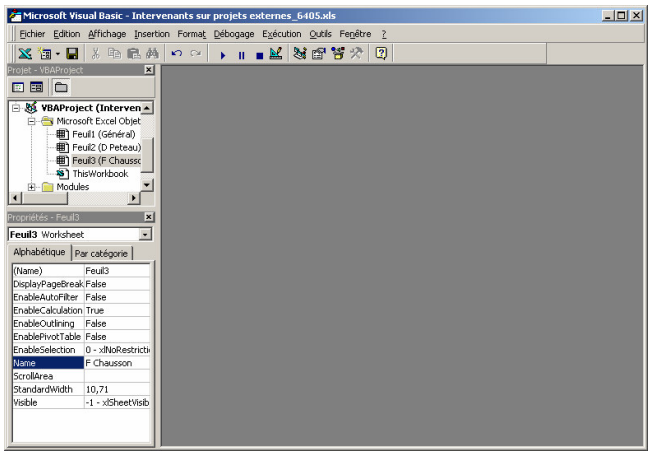
Nous verrons plus loin comment associer vos macros à des boutons sur une barre d'outils ou sur la feuille de calcul ou les incorporer à des menus.

Avec l'Editeur de macros

Vous venez de créer une macro avec l'enregistreur de macros. Vous pouvez la lancer en tapant Ctrl + b. Mais vous ne voyez pas encore à quoi elle « ressemble ». Cette dernière est stockée, dans le classeur Excel, dans une zone spéciale en marge des feuilles. C'est le moment d'y entrer.

Ouvrez l'éditeur Visual Basic

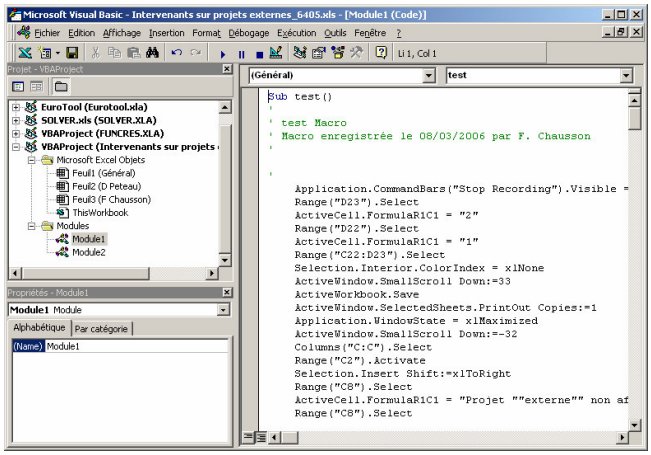
Dans votre feuille de calcul, déroulez le menu **Outils, Macro, Visual Basic Editor** ou tapez le raccourci **Alt + F11.** Vous ouvrez l'Editeur Visual Basic (nous l'appellerons désormais **Editeur).**



Bien qu'incorporé à Office, cet éditeur est un logiciel à part. Vous pouvez donc à tout moment passer d'Excel à l'Editeur et vice-versa en utilisant le raccourci clavier **Alt + Tab.**

Découvrez la fenêtre de l'Editeur

Dans le volet de gauche, apparaissent plusieurs dossiers nommés **VBAProject...** Cliquez sur celui qui porte le nom du classeur ouvert. Vous trouverez deux dossiers : **Microsoft Excel Objet** et **Modules.** Chacun d'eux est précédé par une icône de dossier et un petit signe. Comme dans l'Explorateur de Windows, ce petit signe (un « - » ou un « + ») permet d'afficher ou de masquer le contenu de ce dossier. Grâce à ces deux dossiers vous pourrez choisir où stocker les macros :



Assurez-vous que le dossier **Modules** est ouvert et double-cliquez sur **Module1.** C'est toujours dans ce dossier **Module1** (sauf indication contraire) que vous écrirez vos nouvelles macros.

Regardez le code.

Dans le volet de droite de **l'Editeur** , vous voyez apparaître le code, c'est ainsi que l'on désigne la liste des instructions de votre macro. Ici, elles ont été écrites par l'enregistreur. Vous allez y trouver quelques éléments, communs à toutes macros.

Un nom

Toute macro commence par l'instruction **Sub Nnnn()** - **Nnnn** étant le nom de la macro - et se termine par l'instruction **End Sub.** Ce nom ne doit pas comporter d'espaces.

Des instructions

Elles sont exécutées, sauf ordre contraire, une fois chacune, de haut en bas. Vous pouvez placer plusieurs instructions sur la même ligne, à condition de les séparer par le caractère deux-points (:).

Des commentaires

Toutes les lignes qui commencent par une apostrophe, par exemple **' Macro enregistrée le Nnn par X xxx,** ne sont que des annotations. Elles n'ont aucune influence sur le déroulement de la macro. Elles sont néanmoins utiles, pour vous comme pour des tiers, pour en expliquer le but ou en détailler des portions complexes.

L'Editeur affiche en vert ces lignes de commentaires, mais vous pouvez changer cette couleur, ou *« aérer »* une macro en y laissant des lignes blanches.

Optimisez la macro.

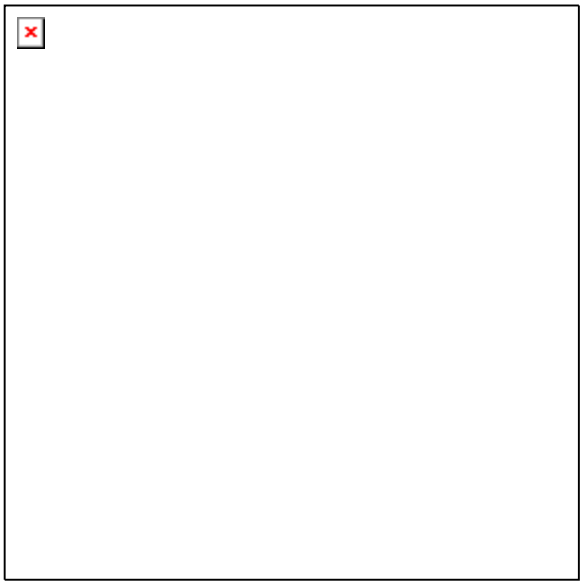
C'est là un point essentiel à comprendre : les macros que crée **l'Enregistreur** sont truffées d'instructions inutiles. Ainsi, sur la douzaine de lignes entre le **Sub ...** et le **End Sub, seules** trois sont utiles !

Vous allez donc, sans quitter **l'Editeur,** créer une nouvelle macro, que vous nommerez **FormaterMieux** et qui fera la même chose que la macro **Formater** , mais qui ne contiendra que les instructions nécessaires.

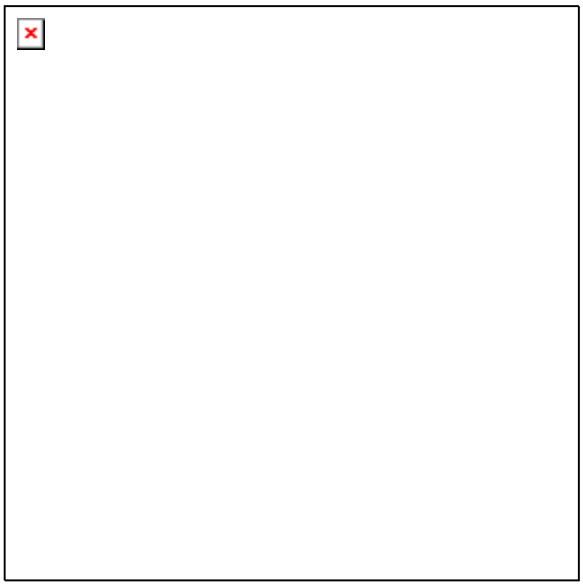
Placez le point d'insertion à la fin du texte des macros en tapant **Ctrl + Fin** , puis **Entrée** pour passer à la ligne. Saisissez la première ligne de votre nouvelle macro, soit **Sub FormaterMieux()** et appuyez sur **Entrée** . L'Editeur insère automatiquement l'instruction **End Sub,** la dernière ligne de votre macro, et ajoute un trait horizontal pour la séparer de la précédente.

Recopiez à présent les trois instructions suivantes :

Selection. Font. Bold = True Selection



Font. Italic = True Selection



HorizontalAlignment = xlCenter

Remarquez qu'Excel colore en bleu les mots-clés reconnus (par exemple **Font, True...).** Vous pouvez ainsi identifier les fautes de frappe.
Autre aide efficace : **l'Editeur** complète automatiquement les instructions que vous tapez en proposant, par exemple, la liste des commandes autorisées pour tel ou tel mot-clé. La première instruction demande à Excel, dans les cellules sélectionnées **(Selection),** de modifier la police **(Font)** et de donner à la propriété Gras **(Bold)** la valeur Vrai **(True).** Notez que l'on décrit les éléments du général vers le particulier.

La seconde instruction active les caractères italiques, et la troisième donne à la propriété Alignement horizontal **(Horizontal Alignment)** de la sélection la valeur Centré **(xlCenter).**

Désactivez la sécurité antivirus.

Par défaut, le système de protection d'Excel contre les macrovirus est placé sur le mode de protection maximal. Pour exécuter vos propres macros sans subir de continuels messages d'avertissements, revenez sous Excel en cliquant sur le bouton **Affichage Microsoft Excel** de la barre d'outils de l'Editeur. Déroulez alors le menu **Outils, Macro, Sécurité,** cochez l'option **Basse** et validez par **OK.**

Exécutez la macro.

Vous n'avez pas attribué de raccourci clavier à **FormaterMieux.** Pour lancer cette macro, sélectionnez une plage de cellules, déroulez **Outils, Macro, Macros** (ou tapez **Alt + F8).** Dans la fenêtre qui s'ouvre, vous voyez s'afficher **Formater,** la macro créée avec l'Enregistreur et **FormaterMieux,** celle écrite avec l'Editeur. Pour en lancer une, sélectionnez-la et cliquez sur **Exécuter.** Notez qu'elles font la même chose.

Gérer ses macros

Vous connaissez maintenant le principe de création et d'enregistrement des macros. Il vous reste à apprendre à les organiser pour les rendre facilement accessibles.

Créez des macros universelles

Normalement, les macros ne sont utilisables que dans le classeur où vous les avez créées. Mais vous pouvez en concevoir qui soient utilisables dans n'importe quel classeur. Pour cela, vous devez construire votre classeur de macros personnelles.

Lancez Excel et déroulez Outils, Macro, Nouvelle macro. Donnez un nom à la macro et, dans la liste Enregistrer la macro dans... choisissez Classeur de macros personnelles.

Effectuez les actions à enregistrer et interrompez l'enregistrement. Le tableur crée, sauf s'il existe déjà, un classeur nommé Perso.xls et le place dans C:\Documents and Settings\\Application Data\Microsoft\Excel\XLSTART.

Attention : ce classeur possède l'attribut Caché (Perso.xls ne figure donc pas parmi les options par défaut de l'Explorateur). Il sera ouvert automatiquement dès que vous lancerez Excel et les macros contenues seront utilisables dans n'importe quel classeur.

Pour élaborer ce classeur, vous avez utilisé l'Enregistreur de macros. Mais vous pouvez écrire vous-même vos macros avec l'Editeur. Pour ajouter une macro universelle nommée Mon Exemple , ouvrez l'Editeur (Alt + F11).

Dans le volet de gauche, déroulez VBAProject (Perso.xls), Modules, Module1 et double-cliquez sur Module1.

Dans le volet de droite, tapez Sub MonExemple() en fin de texte. Entrez les instructions de la macro avant le End sub ajouté par l'Editeur, en vous inspirant des sections suivantes.

Lancez facilement vos macros

Pour exécuter vos macros, vous pouvez utiliser le menu Outils, Macro, Macros. Mais il y a des moyens plus agréables... et plus rapides.

Ajoutez un bouton dans la barre d'outils.

Déroulez le menu Outils, Personnaliser. Dans la liste Catégories, choisissez Macros.

Dans la liste Commandes, cliquez sur Bouton personnalisé et faites glisser ce bouton sur une barre d'outils. Effectuez ensuite un clic droit sur ce nouveau bouton, choisissez Modifier l'image du bouton, cliquez sur l'icône de votre choix puis sur OK.

La première fois que vous cliquerez sur le nouveau bouton, le programme vous demandera quelle macro lui associer.

Placez un bouton dans la feuille de calcul

Déroulez Affichage, Barre d'outils et cochez Formulaires.

Cliquez sur le bouton Bouton et dessinez à la souris un rectangle sur la feuille. La fenêtre Affecter une macro s'ouvre automatiquement. Choisissez la macro à associer au bouton et validez par OK.

Vous pouvez saisir n'importe quel texte à l'intérieur du bouton (le nom ou le rôle de la macro, par exemple). Pour cela, cliquez une fois dans le bouton, tapez le texte et cliquez en dehors pour le valider.

Pour supprimer le bouton, faites un clic droit dessus et, dans le menu contextuel, optez pour Couper.

Associez un raccourci clavier

Déroulez Outils, Macro, Macros. Sélectionnez une macro et cliquez sur Options. Dans le champ Touche de raccourci, tapez un caractère, par exemple m. Cliquez sur OK. Puis fermez la fenêtre Macros.

Vous pouvez maintenant lancer votre macro en tapant Ctrl + m. Evitez d'employer les raccourcis clavier déjà utilisés par Excel, par exemple Ctrl + V, associé à la commande Edition, Coller ou Ctrl + A qui sélectionne toute la feuille.

Ecrire des macros simples

Pour vous aider à faire vos premiers pas en VBA, nous allons débuter par une série d'exemples de macros mettant en oeuvre des concepts élémentaires de programmation.

Modifiez des attributs de cellules

Ce que fait la macro : Elle met en gras et en italique les cellules sélectionnées dans une feuille Excel.

Préparation : Dans une feuille Excel, sélectionnez une plage de cellules.

```
Sub GrasItalique ()

Selection.Font.Bold =True

Selection.Font.Italic=True

End Sub
```

- Explications

```
Selection.Font.Bold=True
```

Dans cette instruction, l'objet Selection représente les cellules sélectionnées dans votre feuille Excel. Vous allez modifier l'attribut Font (polices) de cet objet. Cet attribut Font possède lui-même des sous-attributs, comme Bold (caractères gras) et Italic (italique). Vous donnerez à ces deux attributs la valeur True (vrai).

Remarque : Vous pouvez obtenir des explications sur la syntaxe d'un mot-clé ou d'une commande. Pour cela, sélectionnez ce mot-clé dans l'Editeur et appuyez sur Ctrl + F1.

Recopiez une cellule

Ce que fait la macro : Dans la feuille courante, elle recopie la cellule sélectionnée dans la cellule A1.

Préparation : Ouvrez une feuille Excel et sélectionnez une cellule quelconque contenant des données.

```
Sub Copie ()
```

```
Selection.Copy

ActiveSheet.Paste Destination := Range ( « A1 » )

End Sub
```

- Explications

L'instruction Selection.Copy place dans le Presse-papiers de Windows la zone sélectionnée. C'est l'équivalent de la commande Edition, Copier.

L'instruction suivante copie (Paste) ce contenu dans la feuille active (ActiveSheet) à l'emplacement A1.

L'instruction Range, permet de préciser la destination de cette copie.

Remarque : Notez le caractère := dans l'instruction Range.

Changez des attributs de cellules

Ce que fait la macro : La même chose que la macro GrasItalique.

Préparation : Dans une feuille Excel, sélectionnez une plage de cellules.

```
Sub GrasItaliqueMeilleur()

With Selection.Font.Bold = True

. Italic = True

End With

End Sub
```

- Explications

Par rapport à la macro GrasItalique, vous constatez une différence : la présence d'une instruction With Selection.Font. Cette structure With (avec) ouvre une suite d'instructions qui se termine toujours par l'instruction End With.

Dans toutes les instructions placées entre le With et le End With, l'argument indiqué dans le With (ici Selection.Font) est implicite. En d'autres termes, les deux instructions . Bold=True et .Italic=True concernent toutes les deux l'objet Selection.Font.

Cette façon d'écrire permet de réduire la taille des macros volumineuses en ne citant qu'une seule fois (en « mettant en facteur » , diraient les matheux) les éléments communs à plusieurs instructions.

Remarque : N'oubliez pas les points placés au début des lignes 3 et 4. Ils permettent de séparer les différents arguments.

Ajoutez 15 % à toutes les cellules sélectionnées

Ce que fait la macro : Elle multiplie par 1,15 le contenu de toutes les cellules de la sélection.

Préparation : Dans une feuille Excel, sélectionnez une plage de cellules contenant des nombres.

```
Sub Multiplier1 ()

For Each MaCellule In Selection

MaCellule.Value = MaCellule. Value * 1.15

Next MaCellule

End Sub
```

- Explications

Vous découvrez ici la notion de boucle, qui commence avec For Each MaCellule et qui se termine avec Next MaCellule.

Toutes les instructions placées dans la boucle (ici, il n'y a qu'une seule instruction MaCellule.Value...) seront appliquées successivement à toutes les cellules de la sélection.

Quant à la multiplication proprement dite, elle est effectuée par l'instruction MaCellule.Value = MaCellule.Value * 1.15. Ici, le signe égal doit être compris comme « devient égal à ». En d'autres termes, la valeur (propriété Value) de chaque cellule devient son ancienne valeur multipliée par 1,15.

Remarque : Utilisez toujours le point comme séparateur décimal (1.15 et non 1,15) dans les listings de macros, même si vous employez la virgule dans les feuilles de calcul.

Fusionnez une cellule sur plusieurs lignes

Ce que fait la macro : Elle fusionne le contenu des cellules d'une colonne sur plusieurs autres colonnes contiguës.

Préparation : Ouvrez une feuille Excel et sélectionnez une plage dans laquelle la colonne de gauche contient des données et les autres colonnes sont vides. Lancez la macro. Chaque texte de la colonne de gauche est centré sur toute la largeur des colonnes sélectionnées. C'est l'équivalent de ce que donne le bouton Fusionner de la barre d'outils Mise en forme, mais appliqué à plusieurs lignes.

```
Sub MultiCentrer()

Selection.HorizontalAlignment = xlCenterAcross Selec tion

End Sub
```

- Explications

Une seule instruction, ici, donne à la propriété HorizontalAlignment (alignement horizontal) de l'objet Selection la valeur xlCenterAcrossSelection (centrer sur toutes les colonnes de la sélection).

Cette valeur est une constante prédéfinie par Excel. Toutes ces constantes ont un nom qui commence par xl.

Ajoutez un pourcentage à toutes les cellules sélectionnées

Ce que fait la macro : Elle ajoute à toutes les cellules de la sélection un pourcentage demandé à l'utilisateur en début de macro.

Préparation : Dans une feuille Excel, sélectionnez une plage de cellules contenant des nombres.

```
Sub Multiplier2()

Dim Taux as Single Taux = InputBox (« Taux d'augmentation »)

For Each Cellule In Selection

Cellule.Value = Cellule.Value * (1 + Taux / 100)

Next Cellule

End Sub
```

- Explications

Cette macro constitue une version améliorée de la macro Multiplier1, car elle ajoute de l'interactivité. Nous utilisons ici une variable. Il s'agit d'un emplacement en mémoire, désigné par un nom (Taux) et possédant un type donné et un contenu. Cette variable sert à mémoriser des variables intermédiaires. Le nom d'une variable est libre mais ne doit pas contenir d'espaces.

L'instruction Dim Taux as Single demande à Excel de créer une variable numérique nommée Taux. Le type Single (simple) précise qu'il s'agit d'une variable numérique en simple précision.

Pour avoir un aperçu de tous les types de variables existants (texte, date...), tapez Types de variables dans l'onglet Aide intuitive de l'aide de VBA. Vous constaterez qu'on doit, en l'occurrence, éviter d'utiliser le type Integer qui, comme son nom l'indique, ne peut recevoir que des nombres entiers. Vous ne pourriez plus, alors, saisir un taux de 3,5. L'instruction Taux = InputBox (« Taux d'augmentation ») ouvre une fenêtre de dialogue qui permet à l'utilisateur de saisir un nombre, affecté à la variable Taux. Enfin, au lieu de multiplier par 1,15 comme dans la macro Multiplier1, nous multiplions ici par 1+Taux/100.

Remarque : Dans certains cas, la déclaration de variables n'est pas obligatoire. Nous vous conseillons néanmoins de le faire systématiquement en début de macro car cela rend votre listing plus lisible. Attention : pour déclarer deux variables Var1 et Var2 de type Single , ne

tapez pas Dim Var1, Var2 as Single, mais saisissez deux instructions séparées (sur deux lignes) : Dim Var1 as Single et Dim Var2 as Single.

Ajoutez un pourcentage donné à certaines cellules de la sélection

Ce que fait la macro : Elle ajoute, à toutes les cellules de la sélection dont le contenu est inférieur à 100, un pourcentage demandé à l'utilisateur en début de macro.

Préparation : Dans une feuille Excel, sélectionnez une plage de cellules contenant des nombres.

```
Sub Multiplier3 ()

Dim Taux as Single Taux = InputBox (« Taux d'augmentation »)

For Each Cellule In Selection

If Cellule.Value < 100 Then

Cellule.Value = Cellule. Value * (1 + Taux / 100)

End If

Next Cellule

End Sub
```

- Explications

Par rapport à la macro Multiplier2, ce listing apporte une notion fondamentale : les instructions conditionnelles, qui ne s'exécutent que si une condition est vérifiée. Le principe est le suivant :

Une structure conditionnelle commence avec une instruction if, suivie d'une condition (il s'agit en général d'une comparaison entre deux variables ou constantes, par exemple if A = 5 ou if PrixCourant > 100). A chaque if, doit obligatoirement correspondre une instruction end if qui marque la fin des instructions conditionnelles.

Les instructions placées entre le if et le end if ne sont réalisées que si la condition est vérifiée. Dans notre exemple, la condition est Cellule.Value < 100 . La multiplication ne s'applique qu'aux cellules de la sélection dont le contenu est inférieur à 100.

Remarque : Il est parfois nécessaire de prévoir des choses à faire si la condition est vraie ... et d'autres si la condition est fausse. Vous utiliserez pour cela, les instructions if ... then ... else... end if.

Par exemple, pour doubler les cellules inférieures à 100 et tripler les autres, vous utiliserez le listing suivant :

```
If Cellule.Value < 100 Then
```

```
Cellule.Value = Cellule.Value * 2

Else

Cellule.Value = Cellule. Value * 3

End if
```

Vous pouvez l'interpréter ainsi :

SI (if) la valeur de la cellule est inférieure à 100, ALORS (then) la doubler SINON (else) la tripler. FIN (end if).

Inversez l'attribut Gras de toutes les cellules sélectionnées

Ce que fait la macro : Dans toutes les cellules sélectionnées, elle met en caractères gras les cellules qui ne l'étaient pas et réciproquement.

Préparation : Dans une feuille Excel, sélectionnez une plage de cellules dont certaines sont en caractères gras.

```
Sub InverseGras1 ( )

For Each Cellule In Selection

If Cellule.Font.Bold Then

Cellule.Font.Bold = False

Else

Cellule.Font.Bold = True

End If

Next Cellule

End Sub
```

- Explications

Comme la macro Mutliplier3, nous utilisons ici une instruction conditionnelle : if Cellule.Font.Bold .

Mais la valeur Bold (caractères gras) est de type logique (on dit aussi booléen), ce qui signifie qu'elle ne peut prendre que deux états possibles Vrai ou Faux.

Dans ces conditions, VBA vous permet de réduire l'instruction if Cellule.Font.Bold = True (si Cellule. Police. gras = Vrai) en if Cellule.Font.Bold (si Cellule. Police. gras).

Inversez l'attribut Gras de toutes les cellules sélectionnées

Ce que fait la macro : Elle fait exactement la même chose que la macro InverseGras, mais un peu plus vite.

Préparation : Dans une feuille Excel, sélectionnez une plage de cellules dont certaines sont en caractères gras.

```
Sub InverseGras2 ( )

For Each Cellule In Selection

Cellule.Font.Bold = not Cellule.Font.Bold

Next Cellule

End Sub
```

- Explications

Ici, nous n'utilisons pas de test pour déterminer si une cellule a l'attribut caractère gras. Comme cet attribut est de type logique ou booléen, il ne peut donc avoir deux états, Vrai ou Faux. L'instruction Cellule. Font. Bold = not Cellule. Font. Bold se contente donc d'inverser, via l'opérateur not, l'état de cet attribut. L'absence de if... end if permet à la macro de s'exécuter un peu plus vite que la macro InverseGras.

Supprimez tous les noms dans une feuille de calcul

Ce que fait la macro : Elle efface tous les noms de champs que l'on a créés dans une feuille de calcul. Les champs eux-mêmes ne sont pas effacés.

Préparation : Lancez Excel et ouvrez un classeur, puis activez une feuille contenant des noms de champs.

```
Sub SupprimeTousNoms ()

Dim CeNom as Object

For Each CeNom In ActiveWorkbook. Names

CeNom. Delete

Next CeNom

End Sub
```

- Explications

CeNom est ici une variable objet, qui représente, un par un, tous les éléments de la collection ActiveWorkbook.Names (noms de champs de la feuille active). A chacun de ces noms, on applique la méthode Delete (effacement).

Elaborer des macros évoluées

Avec les exemples qui suivent, vous pourrez aborder des notions de programmation avancées qui vous permettront de créer des macros plus élaborées.

Renommez tous les onglets d'un classeur

Ce que fait la macro : Elle donne à tous les onglets du classeur ouvert un nom composé d'un texte générique, demandé au début à l'utilisateur, et d'un numéro différent pour chaque feuille.

```
Sub RenommeOnglets ()

Dim Z As Integer

Dim Nom As String

Nom = InputBox (« Quel est le nom générique »)

For Z = 1 To Worksheets.Count

Worksheets (Z).Name = Nom & Z

Next Z

End Sub
```

- Explications

Les premières instructions servent à déclarer les variables de travail : Z est un compteur, nous y reviendrons.

La variable Nom, de type texte (string) recevra le nom commun à tous les classeurs.

L'instruction Nom = InputBox (« Quel est le nom générique ») ouvre une fenêtre dans laquelle l'utilisateur saisit un nom qui sera affecté à la variable Nom .

Nous utilisons ensuite une boucle For... Next pour balayer toutes les feuilles du classeur. Il s'agit d'une structure dite de « boucle avec compteur » : toutes les instructions placées entre for Z = ... et Next Z sont exécutées pour toutes les valeurs successives de Z .

C'est l'instruction for Z = qui fixe les valeurs limites. Ici, la boucle donnera à Z toutes les valeurs comprises entre 1 et Worksheets.Count, une variable système qui contient le nombre de feuilles dans le classeur. Ainsi, si le classeur contient 5 feuilles, l'instruction suivante sera exécutée 5 fois.

Worksheets(Z). Name = Nom & Z.

Ici, Worksheets(Z). Name représente le nom de la feuille numéro Z . Cette feuille reçoit un nom formé du nom générique que vous avez tapé, suivi du numéro de feuille. Ainsi, si vous avez choisi le nom Factures , vos feuilles seront successivement nommées Facture1, Facture2, Facture3, Facture4...

Remarque : Le caractère « & » de concaténation : il permet de juxtaposer du texte (la variable caractère Nom) et un nombre (la variable numérique Z), une souplesse rare dans les langages de programmation.

Ajoutez un commentaire indiquant le prix TTC

Ce que fait la macro : Toutes les cellules sélectionnées (elles contiennent un prix hors taxes) reçoivent un commentaire (il apparaîtra dans une infobulle quand vous placerez le pointeur sur les cellules). Cette explication contient le contenu de la cellule, multiplié par 1,196, autrement dit augmenté de 19,6 % (ce qui transforme un prix HT en prix TTC).

Préparation : Dans une feuille, sélectionnez une plage de cellules contenant des prix hors taxes. Après exécution de la macro, chaque cellule apparaît avec une petite marque rouge en haut à droite qui indique la présence d'un commentaire.

```
Sub Taxes ()

Dim TTC As Currency

Dim Cellule As Object

Selection.ClearComments

For Each Cellule In Selection

Cellule.AddComment

TTC = Cellule.Value * 1.196

Cellule.Comment.Text « soit TTC « & Format(TTC, « # 0.00 » ) Next Cellule End Sub
```

- Explications

La première instruction déclare une variable Taxes de type Currency (unité monétaire). Ce type est celui qui offre la meilleure précision pour les montants financiers : les variables de ce type peuvent contenir jusqu'à 15 chiffres avant la virgule et quatre après, sans aucun arrondi. Avant d'attribuer des commentaires aux cellules, nous devons nous assurer que tous les éventuels commentaires existants sont effacés (Selection.ClearComments).

Dans la boucle, l'instruction Cellule.AddComment attribue un commentaire à l'objet nommé Cellule. La variable TTC reçoit ensuite la valeur toutes taxes comprises, soit la cellule multipliée par 1,196.

Enfin, le texte du commentaire de cette cellule (Comment.Text) reçoit une phrase formée des mots « soit TTC » suivis du nombre précédemment calculé, auquel on a appliqué la fonction Format. Le rôle de cette dernière est de contrôler l'affichage d'un nombre, comme vous le faites avec le menu Format, Cellules, Nombres dans Excel.

Ici, l'argument « #0.00 » (avec un point, pas une virgule, attention !) demande un affichage du prix TTC avec deux chiffres décimaux. Après exécution de la macro, si vous passez le pointeur sur une cellule contenant par exemple 500, vous lirez le commentaire « soit TTC

598,00 » . Dans le commentaire, c'est bien une virgule qui apparaît, car cela dépend des paramètres de Windows.

Insérez une ligne sur deux dans un tableau

Ce que fait la macro : Dans une feuille de calcul, elle insère une ligne blanche entre chacune des lignes du tableau.

Préparation : Ouvrez une feuille de calcul et assurez-vous que le tableau à formater commence en A1.

```
Sub InsereUneLigneSurDeux()

Dim Ligne As Integer

Range ( « A1 » ). Select

Ligne = 1

Boucle: Ligne = Ligne + 2

Rows (Ligne).Select

Selection.Insert Shift:=xlDown

If Ligne < ActiveSheet.UsedRange.Rows.Count Then

GoTo Boucle

End If

End Sub
```

- Explications

La macro commence par sélectionner la cellule A1.

La ligne Boucle: n'est pas une instruction, mais une simple étiquette, nous y reviendrons.

L'instruction Rows (Ligne). Select sélectionne la ligne entière.

Puis la macro insère une ligne (Insert) après la ligne courante (Shift:=xlDown).

Le test qui suit décide s'il faut continuer la macro. Si la ligne courante (Ligne) est inférieure au nombre total de lignes dans le tableau (donné par la variable ActiveSheet.UsedRange.Rows.Count), l'exécution de la macro se poursuit (en passant par la commande GoTo) à l'étiquette Boucle: placée plus haut, sinon elle se termine.

Remarque :

L'usage du GoTo est déconseillé dans des programmes de grande taille, car il rend les listings difficiles à comprendre. Toutefois, dans la mesure où l'étiquette et le GoTo qui y renvoient sont proches, cette structure peut quand même vous faire gagner quelques lignes de code.

Sélectionnez toutes les cellules répondant à un critère précis

Ce que fait la macro : Dans une feuille de calcul, elle sélectionne toutes les cellules qui contiennent la même valeur qu'une cellule donnée (la cellule E1 dans notre exemple).

Préparation : Ouvrez une feuille de calcul, mettez la valeur de référence en E1 et saisissez un tableau qui commence en A1.

```
Sub SelectCellulesValeurDonnee ( )

Modele = Range ( « E1 » ). Value

Plage = «»

Range ( « A1 » ).Select

For Each Cellule In ActiveCell.CurrentRegion

If Cellule. Value = Modele Then Plage = Plage & Cellule. Address () & «, »

Next Cellule

If Len (Plage) > 0 Then Range (Lef t(Plage, Len (Plage) -1)). Select

End Sub
```

- Explications

La macro commence par noter la valeur de la cellule E1, qui servira de modèle.

Puis une boucle classique balaie toutes les cellules dans le tableau courant (ActiveCell.CurrentRegion).

Ce « tableau courant» représente une zone autour de la sélection limitée par une ligne vide ou une colonne vide. L'astuce, ici, consiste à utiliser une variable de type texte, nommée Plage. Pour chaque cellule du tableau courant dont la valeur est identique au modèle, l'adresse de cette cellule est ajoutée au contenu de la variable Plage et on y ajoute un séparateur : le point-virgule.

Par exemple, celle-ci contiendra successivement B3; puis B3;C15; puis B3;C15;E17; et ainsi de suite.

Si au moins une cellule a été sélectionnée (la longueur de Plage n'est donc pas nulle, soit Len(Plage) > 0), on sélectionne une plage de cellules formée par la variable Plage.

Mais il faut, au préalable, en ôter le dernier caractère qui est un point-virgule. C'est le rôle de la fonction Left(Plage, Len(Plage) -1) qui retient, à gauche de la variable, un nombre de caractères égal à la longueur de cette variable moins un.

Ainsi, si la variable Plage contenait B3; C15;E17; c'est bien la plage B3;C15;E17 qui sera sélectionnée.

Remarque : Cette macro comble une lacune d'Excel qui n'offre aucun moyen direct de sélectionner les cellules contenant une valeur donnée.

Eliminez les lignes vides dans un tableau

Ce que fait la macro : Elle supprime, dans toute la feuille, les lignes entièrement vides.

Préparation : Ouvrez une feuille contenant un tableau.

```
Sub SupprimerLignesVides ()

Dim Derniere As Long

Dim Ligne As Long

Dim CellulesOccupees As Integer

Derniere = ActiveSheet. UsedRange. Rows. Count

Application. ScreenUpdating = False

Ligne = Derniere

For Ligne = Derniere To 1 Step -1

CellulesOccupees = Application. CountA(Rows (Ligne))

If CellulesOccupees = 0 Then Rows (Ligne). Delete

Next Ligne

End Sub
```

- Explications

La macro commence par déterminer le nombre de lignes dans la « zone utile ». Ce nombre est donné par la variable d'environnement ActiveSheet.UsedRange.Rows.Count (feuille active. Zone utilisée. Lignes. Nombre).

Pour avoir une idée de ce qu'Excel considère comme zone utile, sélectionnez la cellule A1 et tapez Ctrl + Fin. L'instruction Application.ScreenUpdating = False est facultative : elle permet à la macro de s'exécuter un peu plus vite en désactivant la mise à jour de l'affichage entre chaque instruction.

Une boucle for... next va ensuite balayer le tableau en partant de la dernière ligne.

Pour supprimer des éléments dans un tableau, il est plus facile de le parcourir de bas en haut. C'est pour forcer un comptage décroissant que l'on ajoute la clause step -1 dans l'instruction for (incrément de -1).

La macro compte ensuite, en exploitant la fonction CountA, le nombre de cellules non vides dans la ligne en cours. Si ce nombre est nul, la ligne est supprimée.

Appliquez un formatage conditionnel à six couleurs

Ce que fait la macro : Elle applique, à toutes les cellules sélectionnées, une couleur choisie parmi six, en fonction de la valeur numérique de la cellule, alors que la fonction Formatage conditionnel d'Excel est limitée à trois conditions.

Préparation : Ouvrez une feuille de calcul et sélectionnez une plage de cellules contenant des nombres compris entre 1 et 40 (ou modifiez les valeurs limites des « bornes » dans la macro).

```
Sub Conditionnel ()

For Each Cellule In Selection

Select Case Cellule. Value

Case Is < 11

Cellule. Interior. ColorIndex = 3' rouge

Case 11 To 15

Cellule. Interior. ColorIndex = 46' orange foncé

Case 16 To 20 Cellule. Interior. ColorIndex = 45' orange moyen

Case 21 To 25 Cellule. Interior. ColorIndex = 44' orange clair

Case 26 To 30 Cellule. Interior. ColorIndex = 27' jaune

Case Else Cellule. Interior. ColorIndex = 19' jaune clair

End Select

Next Cellule

End Sub
```

- Explications

La structure if ... then ... else convient quand il faut prendre une décision « binaire » (par exemple if A<10). Mais dans notre exemple, la décision est plus complexe puisque nous

voulons que la couleur d'une cellule puisse prendre six valeurs distinctes en fonction de la valeur de la cellule (inférieure à 11, comprise entre 11 et 15, comprise entre 16 et 20...).

Au lieu d'emboîter des if ... then ... end if, ce qui serait peu pratique, nous employons la structure Select Case , qui permet de créer autant de branches que nécessaire.

L'ouverture de cette structure, Select Case Cellule.Value spécifie que les tests suivants porteront sur la variable Cellule.Value . Il reste à créer une « branche » pour chaque plage de valeurs.

Ainsi, Case 11 To 15 s'applique aux valeurs comprises entre 11 et 15. Sous chaque Case, placez les instructions (ici, une seule) à exécuter dans ce cas.

Notez que la dernière branche, nommée Case Else, décrit les instructions à exécuter si aucune des conditions précédentes n'est vérifiée.

Enfin, c'est l'instruction Cellule.Interior.ColorIndex = Nnn qui se charge de donner un fond coloré à chacune des cellules balayées par la boucle. La structure Select Case doit, comme les autres, être refermée. C'est le rôle de la ligne End Select. Remarque : Dans les lignes Cellule.Interior..., nous avons ajouté un commentaire indiquant « en clair » la couleur désignée par la valeur numérique. Dans une ligne de code, vous pouvez ajouter un commentaire en le faisant précéder d'une apostrophe.

Formatez une cellule sur trois dans une colonne

Ce que fait la macro : Dans une colonne zone sélectionnée, la macro donne à une cellule sur N (ce nombre est demandé au début) le même format que la première cellule de la sélection.

Préparation : Dans une feuille de calcul, donnez à une cellule d'une colonne le format servant de modèle, sélectionnez cette cellule et plusieurs autres dans la même colonne et lancez la macro.

```
Sub MarquerCellules ()

Dim I as Integer : Dim Frequence as Integer

Selection. Item (1,1). Copy

I = 0

Frequence = CInt(InputBox( « Formater une cellule sur : » ))

For Each Cellule In Selection

If I Mod Frequence = 0 Then

Cellule. PasteSpecial xlPasteFormats

End If

I = I + 1
```

Next Cellule

End Sub

- Explications

La macro commence par copier dans le Presse-Papiers la première cellule de la sélection. L'élément Item (1,1) désigne la cellule placée en haut à gauche (ligne 1, colonne 1) de cette sélection.

Puis la fonction InputBox, dont le résultat est converti en valeur Integer par la fonction Cint, demande à quelle fréquence il faut formater les cellules.

Un compteur (la variable I) est mis à zéro.

Dans la boucle principale, ce compteur est augmenté d'une unité (I = I + 1) pour chaque cellule. On cherche alors le reste de la division de I par Frequence avec la fonction Mod (c'est la fonction modulo, la division entière).

Si ce reste est nul, la variable I est donc un multiple de Frequence. Dans ce cas, on ne fait pas un collage classique, mais un collage spécial (PasteSpecial) pour n'appliquer à la cellule en cours que le format de celle servant de modèle (xlPasteFormat).

Lancez une commande à chaque ouverture d'un classeur

Ce que fait la macro : Chaque fois que vous ouvrez le classeur, elle affiche un message « Chiffres provisoires » (cela pourrait être n'importe quelle autre commande).

Préparation : Attention, pour que cette macro fonctionne, vous devez obligatoirement la placer dans le dossier ThisWorkBook. Dans le volet de gauche de l'Editeur , cliquez deux fois sur cet élément et saisissez votre code dans le volet de droite.

Sub WorkBook _Open ()

MsgBox («Chiffres provisoires»)

End Sub

Explications
Certaines macros ont un nom imposé. C'est le cas ici. Dans la mesure où la macro se nomme WorkBook_Open (classeur_Ouvrir), elle sera automatiquement exécutée chaque fois que vous ouvrez ce classeur. On parle ici de programmation événementielle.

Un « événement », pour VBA, c'est l'ouverture d'un fichier, un clic sur une cellule, un appui sur une touche du clavier... A chaque événement, vous pouvez associer une série de commandes.

Dressez la liste des fichiers d'un dossier

Ce que fait la macro : Elle écrit, dans la colonne A de la feuille courante, la liste des fichiers du dossier spécifié (C:\MonDossier dans notre exemple, mais vous pouvez en indiquer un autre selon vos besoins).

Préparation : Ouvrez ou créez une feuille vierge.

```
Sub ListeDesFichiers ()

Dim I As Long

With Application. FileSearch

. New Search

. FileType = msoFileTypeAllFiles

. LookIn = « C:\MonDossier\ »

. SearchSubFolders = True

. Execute

With. FoundFiles

For I = 1 To. Count

Range ( « A1 » ). Offset(I, 0) =. Item (I)

Next I

End With

End With

End Sub
```

- Explications

Toute la macro est écrite autour d'une boucle Application.FileSearch (recherche de fichiers). L'argument . NewSearch indique qu'il s'agit d'une nouvelle recherche.

L'instruction FileType = msoFileTypeAllFiles spécifie le type de fichiers recherchés (ici tous les fichiers). En remplaçant msoFileTypeAllFiles par msoFileTypeBinders, la macro ne chercherait que les classeurs Excel.

L'argument . LookIn spécifie dans quel dossier effectuer la recherche (C:\MonDossier\), enfin SearchSubFolders = True demande à la macro de parcourir également, le cas échéant, les sous-dossiers de C:\MonDossier.

La clause . Execute indique ensuite ce qu'il faut faire avec les fichiers trouvés. La collection . FoundFile est alors balayée par une boucle for... next dont la limite supérieure est FoundFiles. Count (nombre de fichiers trouvés). Chacun des noms de fichiers est alors écrit dans une cellule distincte.

Pour qu'une macro écrive dans une cellule, il est inutile de la sélectionner. Vous pouvez en effet utiliser l'instruction Offset qui admet deux arguments : un nombre de lignes et un nombre de colonnes.

Ainsi, l'instruction Range(«B2»). Offset(3,4) = 2006 écrit la valeur 2006 dans la cellule F5, soit 3 lignes plus bas et 4 colonnes à droite de la cellule B2.

Chaque nom de fichier, un des éléments du tableau Item(I), est ici écrit dans une cellule décalée de I lignes et de zéro colonne (donc en colonne A) par rapport à la cellule A1.

Ajoutez une ligne dans un fichier texte à chaque impression du classeur

Ce que fait la macro : Chaque fois que vous imprimez l'une des feuilles du classeur, la macro écrit, dans un fichier texte (en l'occurrence, le fichier Utilisation. log placé à la racine du disque C) une ligne qui contient le nom de l'utilisateur et la date.

Préparation : Attention, pour que cette macro fonctionne, il faut obligatoirement, comme la précédente, la placer dans le dossier ThisWorkBook. Dans le volet de gauche de l'Editeur, cliquez deux fois sur cet élément et tapez votre code dans le volet de droite.

```
Private Sub WorkBook _ BeforePrint (Annule A s Boolean)

Open « C:\Utilisation. log » For Append A s # 1

Print # 1, Application. UserName, Now

Close # 1

End Sub
```

- Explications

C'est là un deuxième exemple d'une macro événementielle. Ici, nous créons une macro qui sera automatiquement exécutée avant (Before) chaque commande d'impression (Print).

Quand vous ouvrez un fichier texte avec la commande Open, vous devez indiquer si vous avez l'intention d'y lire (for input) ou d'y ajouter (for append) des données.

Dans ce dernier cas, si le fichier Utilisation.log n'existe pas, il est automatiquement créé. Ce fichier, de type texte, peut être lu avec n'importe quel traitement de texte ou le Bloc-notes de Windows.

Enfin, le #1 est appelé numéro de canal : vous l'utiliserez dès que vous voudrez écrire des données dans ce fichier. C'est ce que fait l'instruction Print #1 : elle écrit dans ce fichier le nom de l'utilisateur (Application. UserName , variable prédéfinie d'Excel) et la date et l'heure (la fonction Now). Le fichier Utilisation.log est ensuite refermé par l'instruction Close.

Conversion Excel Access

Conversion Excel vers Access

http://www.laboratoire-microsoft.org/t/1734/

Assurez vous tout d'abord que vos données sont organisées comme des listes :

- la première ligne de chaque colonne doit contenir l'étiquette de cette colonne
- il doit n'y avoir aucune ligne ou colonne vide dans la liste.

Sélectionnez les données que vous souhaitez convertir.

Dans le menu **Données**, cliquez sur **Convertir** puis suivez les instructions

Annexes

Bibliographie « Utiliser … »

Ces différents documents constituent l'ensemble documentaire ***Utiliser ….***

La liste complète est disponible sur http://fceduc.free.fr/documentation.php.

