Guide de formation avec cas pratiques

Excel 2013 Programmation VBA

Daniel-Jean David

Tsoft EDITEUR

EYROLLES

Création d'un Programme

1

Enregistrement d'une macro

Écriture des instructions VBA : l'Éditeur VBA

Règles fondamentales de présentation

Projets, différentes sortes de modules

Options de projets

Les différentes sortes d'instructions

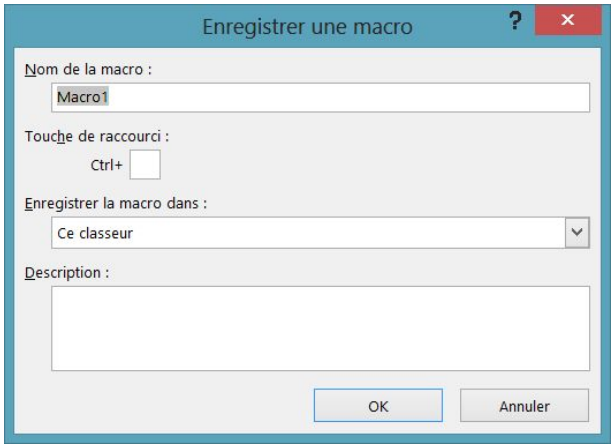
Les menus de l'Éditeur VBA

ENREGISTREMENT D'UNE MACRO

ENREGISTRER UNE SUITE D'OPÉRATIONS EXCEL

Nous allons voir qu'on peut mémoriser une suite d'opérations Excel pour pouvoir répéter cette suite ultérieurement sans avoir à refaire les commandes.

- Dans feuille de classeur Excel, faites 🗀*AFFICHAGE – [Macros] – Macros – Enregistrer une macro :*

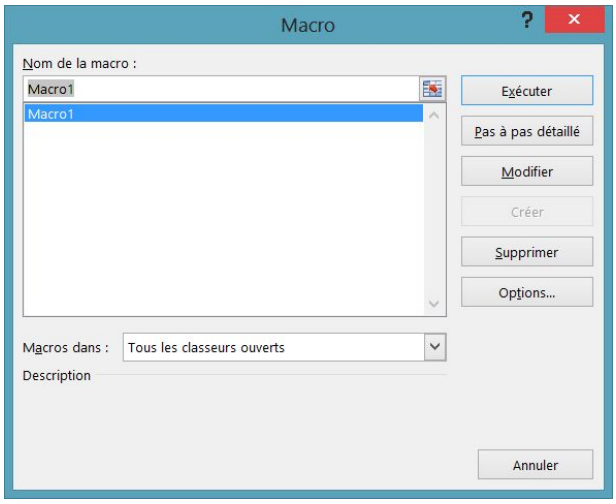


- Vous avez la possibilité de changer le nom de la macro, de la sauvegarder dans d'autres classeurs (le plus souvent, on la sauvegarde dans le classeur en cours) ou de donner une description plus complète de la macro en cours de définition. L'option probablement la plus utile est d'associer une touche de raccourci. Cliquez sur OK pour valider.
- Faites les opérations Excel que vous souhaitez enregistrer….
- Faites 🗀*AFFICHAGE – [Macros] – Macros – Arrêter l'enregistrement*.

Avant l'enregistrement, vous avez la possibilité de demander 🗀*AFFICHAGE – [Macros] – Macros – Enregistrer une macro – Utiliser les références relatives*, ce qui permet de décider que la rédaction de la macro traitera les coordonnées de cellules en relatif (c'est en absolu en l'absence de cette commande).

DÉCLENCHER UNE NOUVELLE EXÉCUTION

- Revenu sur la feuille Excel, modifiez éventuellement certaines données.
- Faites 🗀*AFFICHAGE – [Macros] – Macros – Afficher les macros*, le dialogue suivant s'affiche :



Ce dialogue permet de choisir une macro dans la liste. Cette liste est formée de toutes les procédures connues de Visual Basic soit dans tous les classeurs ouverts, soit dans le classeur spécifié grâce à la liste déroulante <Macros dans> en bas de la BDi.

- Après avoir sélectionné la macro, cliquez sur le bouton Exécuter, vous pouvez constater que vos opérations sont répétées

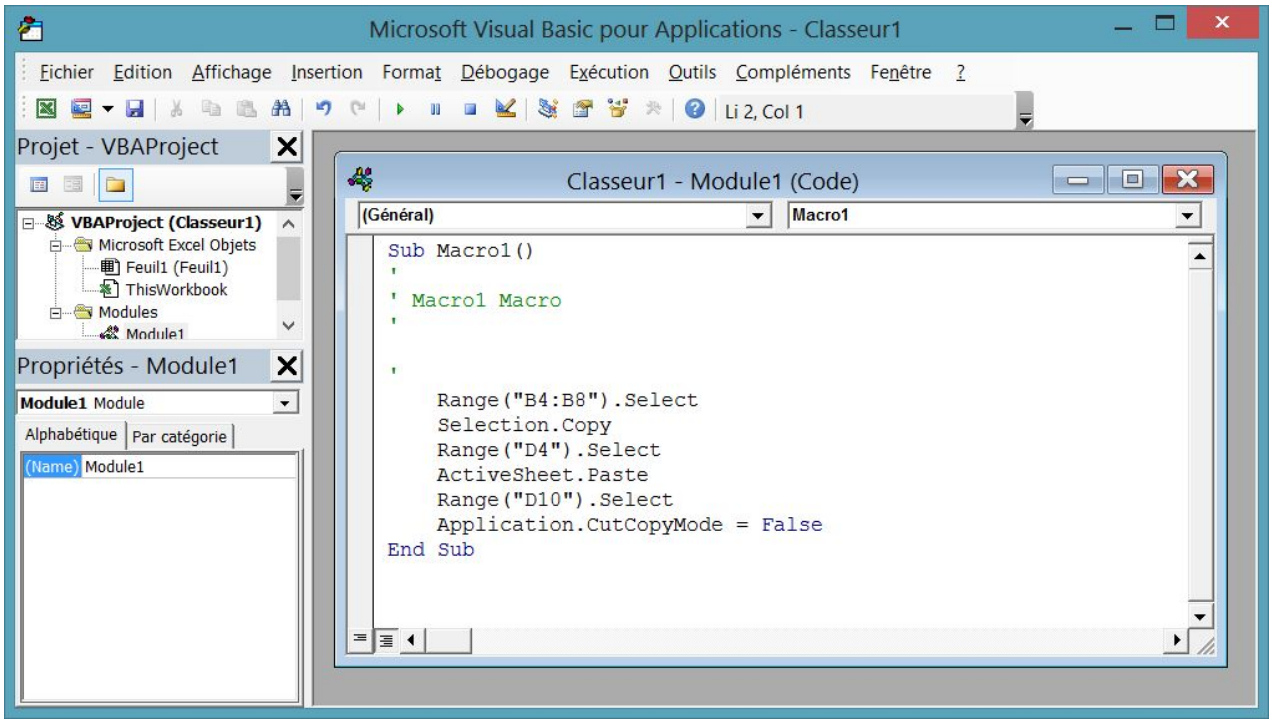
EXAMINER LA MACRO PRODUITE

Il faut pouvoir examiner ce qu'Excel a mémorisé en fonction des actions enregistrées. Cet examen est en particulier nécessaire si l'exécution de la macro ne produit pas les résultats voulus : c'est probablement qu'une action parasite a été enregistrée et il faudra enlever ce qui la représente dans l'enregistrement

Une autre raison d'examiner la macro telle qu'elle est enregistrée est de pouvoir la modifier. Des modifications mineures qu'on peut vouloir faire viennent du processus même de l'enregistrement : supposons que, voulant sélectionner la cellule A3, vous sélectionniez d'abord, suite à une hésitation, la cellule A4 ; bien entendu, vous allez rectifier et cliquer sur A3. Mais Excel aura enregistré deux opérations de sélection et il sera conseillé de supprimer la sélection de A4. Donc une première raison de modification est d'élaguer la macro des opérations inutiles.

Un autre motif de modification, beaucoup plus important, est de changer le comportement de la macro pour le rendre plus ergonomique, ou pour traiter d'autres aspects de l'application.

- Dans la boîte de dialogue 🗀*AFFICHAGE – [Macros] – Macros – Afficher les macros*, cliquez sur Modifier : la fenêtre de l'Éditeur VBA apparaît.



L'ONGLET DÉVELOPPEUR

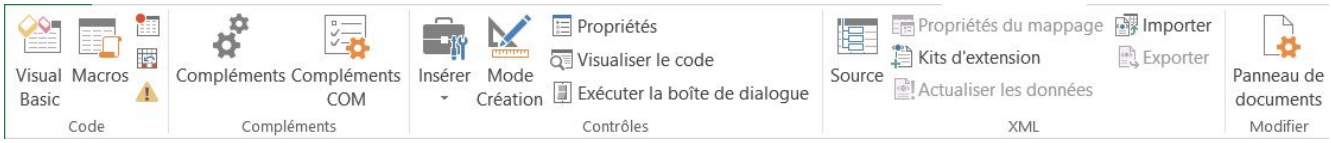
Nous voyons maintenant une autre manière d'appeler l'éditeur VBA. Une option permet d'ajouter un onglet appelé Développeur. Il est, de toutes façons, indispensable pour toute utilisation régulière de VBA.

ENREGISTREMENT D'UNE MACRO

Afficher l'onglet développeur

- Cliquez sur Fichier.
- Cliquez sur Options puis *Personnaliser le ruban*.
- Cochez ☑ Développeur dans la liste *Onglets principaux* et OK .

L'onglet Développeur se rajoute au ruban. Voici son contenu :



La commande 🗀*DÉVELOPPEUR – [Code] – Macros* fait apparaître la boîte de dialogue liste des macros. La commande 🗀*DÉVELOPPEUR – [Code] – Visual Basic* appelle l'éditeur VBA. Vous retiendrez rapidement son raccourci Alt+F11, best-seller auprès des programmeurs VBA.

On passe de la fenêtre de l'éditeur VBA à la fenêtre classeur et inversement par clics sur leurs boutons dans la barre en bas de l'écran ou à coups de Alt+F11.

À part ses barres de menus et d'outils, la fenêtre de l'éditeur VBA comprend deux volets. Celui de gauche se partage de haut en bas en Explorateur de projets et Fenêtre de propriétés ; le volet de droite est occupé par une ou plusieurs fenêtres de code.

- Si vous n'avez pas l'affichage correspondant à la figure, le plus probable est que vous n'ayez pas la fenêtre de code, mais que vous ayez le volet de gauche. Dans l'Explorateur de projets, vous devez avoir au moins une tête d'arborescence *VBAProject(nom de votre classeur)*. Pour VBA, un classeur et l'ensemble de ses macros forme un « projet ». L'arborescence de votre projet doit se terminer par une rubrique Modules.
- Si celle-ci n'est pas développée, cliquez sur son signe + : Module1 doit apparaître
- Double-cliquez sur le mot *Module1* : la fenêtre de code doit apparaître.
- Si vous n'avez pas le volet de gauche, appelez le menu *Affichage* et cliquez les rubriques *Explorateur de projets* et *Fenêtre Propriétés*, puis éventuellement arrangez leurs tailles et positions.

Avantages et inconvénients de la construction de macros par enregistrement

On peut créer une macro sans enregistrer des actions Excel, en écrivant le texte du programme souhaité directement dans une fenêtre module sous l'Éditeur VBA.

Un avantage de l'enregistrement d'une séquence de commandes est que, la macro étant générée par Excel, elle ne peut contenir aucune faute de frappe. Du côté des inconvénients, nous noterons un certain manque de souplesse : la macro ne peut que faire exactement ce qu'on a enregistré, sans paramétrage possible.

Autre inconvénient, plus grave et qui justifie que l'on puisse saisir des programmes directement au clavier : par enregistrement, on ne peut que générer un programme à logique linéaire où toutes les actions se suivent en séquence ; on ne peut pas créer un programme où, en fonction de premiers résultats, on effectue telle action ou bien telle autre : lors de l'enregistrement, on suivra une seule des voies possibles et elle seule sera enregistrée.

A fortiori, lorsqu'une sous-étape du traitement doit être répétée plusieurs fois, l'enregistrement ne mémorise qu'un passage. Ces possibilités appelées « alternatives » et « boucles » sont offertes par des instructions de VBA mais qui doivent être fournies directement. Ces instructions s'appellent instructions de structuration.

Mais un grand avantage de l'enregistrement, qui est à nos yeux le plus important, est que cette méthode est une extraordinaire machine à apprendre VBA, ou plutôt les objets Excel et leur manipulation : dès qu'on sait accomplir une action par les commandes Excel, on saura comment cela s'écrit en VBA, ou plutôt quels objets manipuler et comment. Il suffit de se mettre en mode enregistrement, d'effectuer les commandes Excel voulues, arrêter l'enregistrement puis examiner ce que le système a généré. Par exemple, pour voir comment on imprime, il suffit de commander une impression en mode enregistrement. Bien sûr, on pourrait trouver la réponse dans l'aide en ligne, mais la méthode de l'enregistrement épargne une longue recherche.

Sauvegarde d'un classeur contenant des macros

Bien entendu, votre classeur devra être sauvegardé. Dans la version Office 2013, les classeurs qui ne contiennent pas de macros ont l'extension .xlsx, tandis que ceux qui contiennent des macros ont l'extension .xlsm.

Pour la première sauvegarde du classeur, il faut revenir à la fenêtre Excel et :

- 🗀*FICHIER – Enregistrer sous – Classeur Excel prenant en charge les macros*.
- Fournir disque, répertoire et nom du fichier.

Pour les sauvegardes suivantes, la commande *Fichier – Enregistrer* de la fenêtre de l'éditeur VBA convient.

ÉCRITURE DES INSTRUCTIONS VBA : L'ÉDITEUR VBA

CRÉER UN MODULE

Depuis un classeur Excel, on arrive à l'écran VBA par la commande 🗀*DÉVELOPPEUR – [Code] – Visual Basic* ou Alt+F11. On a vu dans la section précédente comment assurer que la fenêtre de projets soit présente. Elle a au moins une arborescence *VBA Project (nom de votre classeur)* et celle-ci a au moins une rubrique *Microsoft Excel Objects*.

- Si le programme que vous souhaitez écrire doit gérer la réponse à des événements concernant une feuille de classeur ou le classeur, les modules correspondants apparaissent dans l'arborescence sous *Microsoft Excel Objects*. Double-cliquez sur la feuille voulue ou le classeur : la fenêtre de module apparaît.
- Dans les autres cas :
  - Sélectionnez le projet (clic sur sa ligne dans la fenêtre *Projets*), puis
  - *Insertion – Module* pour un module normal. Les autres choix sont *Module de classe* et *User Form* (Boîte de dialogue et module gestion des objets contenus). Ces cas sont traités dans d'autres chapitres, donc plaçons-nous ici dans le cas du module normal.
  - Une fois le module créé, la rubrique *Modules* apparaît dans l'arborescence. Pour écrire le programme, développez la rubrique, puis double-cliquez sur le nom du module voulu.
  - Il faut maintenant créer une procédure. Le menu *Insertion* a une rubrique *Procédure*, mais il suffit d'écrire `Sub <nom voulu>` dans le module.

SUPPRIMER UN MODULE

On peut avoir à supprimer un module, notamment parce que, si on enregistre plusieurs macros, VBA peut décider de les mettre dans des modules différents (par exemple Module2, etc.) alors qu'il est préférable de tout regrouper dans Module 1.

- Après avoir déplacé les procédures des autres modules dans Module 1, sélectionnez chaque module à supprimer par clic sur son nom sous la rubrique Modules.
- *Fichier – Supprimer Module 2* (le nom du module sélectionné apparaît dans le menu *Fichier*).
- Une BDi apparaît, proposant d'exporter le module. Cliquez sur Non.

EXPORTER/IMPORTER UN MODULE

Exporter :

Si dans la BDi précédente, vous cliquez sur Oui, vous exportez le module, c'est-à-dire que vous créez un fichier d'extension .bas qui contiendra le texte des procédures du module. Un tel fichier peut aussi se construire par :

- Mettez le curseur texte dans la fenêtre du module voulu.
- *Fichier – Exporter un fichier*.
- La BDi qui apparaît vous permet de choisir disque, répertoire et nom de fichier.

Importer :

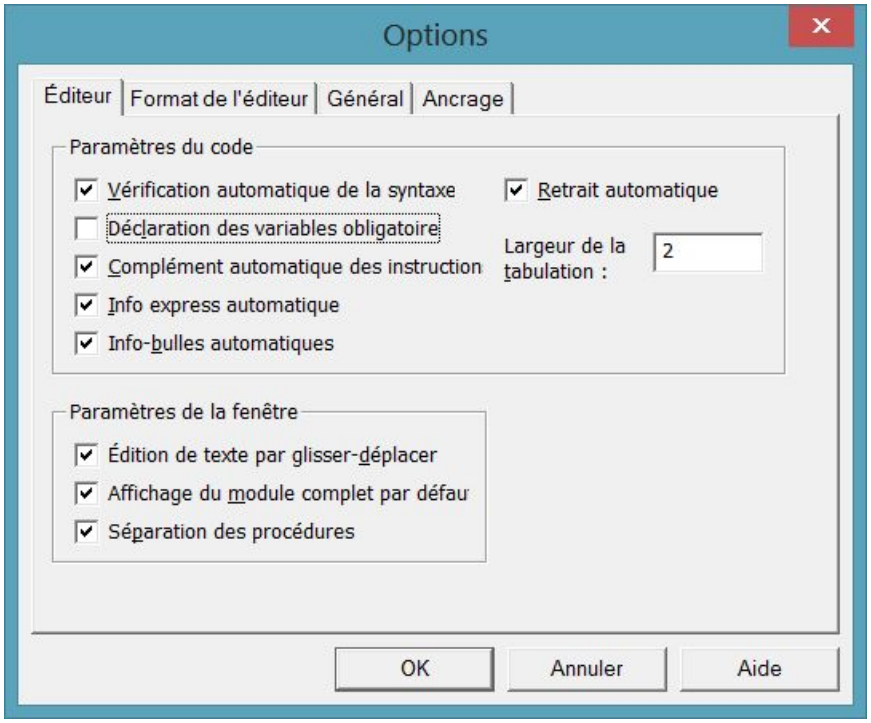
L'opération inverse est l'importation qui permet d'ajouter un fichier à un projet :

- Sélectionnez le projet concerné (par clic sur sa ligne dans la fenêtre de projets), puis faites *Fichier – Importer un fichier*.
- Dans la BDi, choisissez disque, répertoire et nom de fichier. Les extensions possibles sont .bas (module normal), .cls (module de classe) et .frm (BDi construite par l'utilisateur et le module de code associé).

Cette technique permet de développer des éléments, procédures ou BDi servant pour plusieurs projets.

OPTIONS RÉGLANT LE FONCTIONNEMENT DE L'ÉDITEUR

Dans l'écran VBA, faites *Outils – Options*. Le fonctionnement de l'éditeur obéit aux onglets *Éditeur* et *Format de l'éditeur*. L'onglet *Éditeur* règle le comportement vis-à-vis du contenu du programme notamment les aides à l'écriture procurées par l'éditeur :



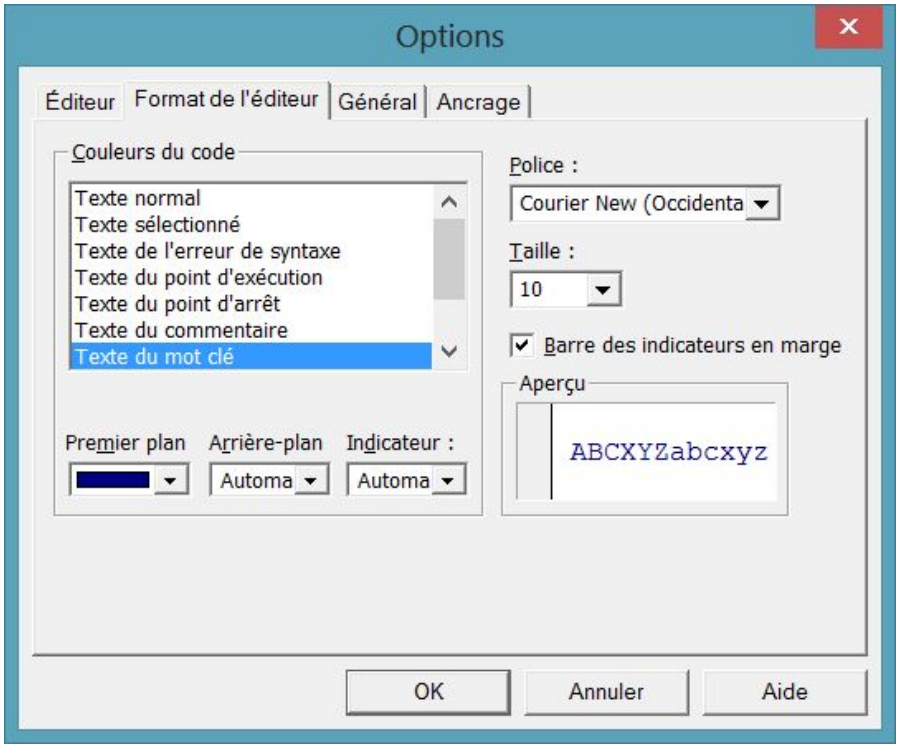
Les choix de la figure nous semblent les plus raisonnables.

- ☑ *Vérification automatique de la syntaxe* parle d'elle-même
- ☐ *Déclaration de variables obligatoire* si la case est cochée installe automatiquement Option Explicit en tête de tous les modules. Si la case n'est pas cochée, vous devez taper la directive partout où il le faut.
- ☑ *Complément automatique des instructions* présente les informations qui sont le complément logique de l'instruction au point où on est arrivé.
- ☑ *Info express automatique* affiche des informations au sujet des fonctions et de leurs paramètres au fur et à mesure de la saisie
- ☑ *Info-bulles automatiques* : en mode Arrêt, affiche la valeur de la variable sur laquelle le curseur est placé.
- ☑ *Retrait automatique* : si une ligne de code est mise en retrait, toutes les lignes suivantes sont automatiquement alignées par rapport à celle-ci. Pensez en même temps à choisir l'amplitude des retraits successifs (ci-dessus 2, au lieu de la valeur par défaut 4).

Les options *Paramètres de la fenêtre* sont moins cruciales.

- ☑ *Édition de texte par glisser-déplacer* permet de faire glisser des éléments au sein du code et de la fenêtre Code vers les fenêtres Exécution ou Espions.
- ☑ *Affichage du module complet par défaut* fait afficher toutes les procédures dans la fenêtre Code ; on peut, par moments, décider d'afficher les procédures une par une.
- ☑ *Séparation des procédures* permet d'afficher ou de masquer les barres séparatrices situées à la fin de chaque procédure dans la fenêtre Code. L'intérêt de cette option est diminué par le fait que ces séparations n'apparaissent pas à l'impression du listing ; une solution est d'insérer devant chaque procédure une ligne de commentaire remplie de tirets : '--------…

L'onglet *Format* de l'éditeur fixe les couleurs des différents éléments du code. C'est lui qui décide par défaut mots-clés en bleu, commentaires en vert, erreurs en rouge.



- ☑ *Barre des indicateurs en marge* affiche ou masque la barre des indicateurs en marge, indicateurs utiles pour le dépannage.
- Ayant choisi un des éléments dans la liste, vous déterminez la police, taille et couleur de façon classique ; en principe, on utilise une police de type Courrier parce qu'elle donne la même largeur à tous les caractères, mais rien ne vous y oblige.
- Les éléments possibles sont : Texte normal, Texte sélectionné, Texte de l'erreur de syntaxe, Texte du point d'exécution, Texte du point d'arrêt, Texte du commentaire, Texte du mot clé, Texte de l'identificateur, Texte du signet, Texte de retour de l'appel.

RÈGLES FONDAMENTALES DE PRÉSENTATION

UNE INSTRUCTION PAR LIGNE

La règle fondamentale est d'écrire une instruction par ligne. Lorsque vous tapez sur la touche ↵, VBA suppose qu'on passe à la prochaine instruction. Cette règle admet deux exceptions qui n'interviennent que très rarement.

- On peut mettre plusieurs instructions sur une ligne à condition de les séparer par le caractère deux-points (:).

```
x = 3 : y = 5
```

Cette pratique est tout à fait déconseillée ; elle ne se justifie que pour deux instructions courtes formant en quelque sorte un bloc logique dans lequel il n'y aura en principe pas de risque d'avoir à insérer d'autres instructions.

- Une instruction peut déborder sur la (les) ligne(s) suivante(s). La présentation devient :

```
xxxxxxxxxxxxxxxxxx(1re partie)xxxxxxxxxxxxxxxxxx   _
        yyyyyyy(2e partie)yyyyyyyyyyyyyyy
```

Les lignes sauf la dernière doivent se terminer par la séquence <espace><signe souligné>. Bien entendu, la coupure doit être placée judicieusement : là où l'instruction aurait naturellement un espace. On ne doit pas couper un mot-clé propre au langage, ni un nom de variable.

Cas particulier : on ne doit pas couper une chaîne de caractères entre guillemets (comme `"Bonjour"`). La solution est la suivante : on remplace la longue chaîne par une concaténation de deux parties ("partie 1" + "partie 2") et on coupera comme suit :

```
 ……."partie 1" +  _
"partie 2"
```

MAJUSCULES ET MINUSCULES

Sauf à l'intérieur d'une chaîne de caractères citée entre ", les majuscules et minuscules ne comptent pas en VBA. En fait, les mots-clés et les noms d'objets et de propriétés prédéfinis comportent des majuscules et minuscules et vous pouvez définir des noms de variables avec des majuscules où vous le souhaitez. Mais vous pouvez taper ces éléments en ne respectant pas les majuscules définies (mais il faut que les lettres soient les mêmes) : l'éditeur VBA rétablira automatiquement les majuscules de la définition ; pour les noms de variables, on se basera sur la 1re apparition de la variable (en principe sa déclaration).

Il en résulte un conseil très important : définissez des noms avec un certain nombre de majuscules bien placées et tapez tout en minuscules : si VBA ne rétablit pas de majuscules dans un nom, c'est qu'il y a une faute d'orthographe.

Un autre élément qui peut vous permettre de déceler une faute d'orthographe, mais seulement dans un mot-clé, est que si un mot n'est pas reconnu comme mot-clé, VBA ne l'affichera pas en bleu. Bien sûr, vous devez être vigilants sur ces points : plus tôt une faute est reconnue, moins il y a de temps perdu.

Pour les chaînes de caractères entre ", il s'agit de citations qui apparaîtront telles quelles, par exemple un message à afficher, le nom d'un client , etc. Il faut donc taper exactement les majuscules voulues.

COMMENTAIRES, LIGNES VIDES

Un commentaire est une portion de texte figurant dans le programme et n'ayant aucun effet sur celui-ci. La seule chose que VBA fait avec un commentaire, c'est de le mémoriser et de l'afficher dans le listing du programme. Les commentaires servent à donner des explications sur le programme, les choix de méthodes de traitement, les astuces utilisées, etc.

Ceci est utile pour modifier le programme, car, pour cela, il faut le comprendre ; c'est utile même pour le premier auteur du programme car lorsqu'on reprend un programme plusieurs mois après l'avoir écrit, on a oublié beaucoup de choses. Il est donc conseillé d'incorporer beaucoup de commentaires à un programme dès qu'il est un peu complexe.

VBA admet des commentaires en fin de ligne ou sur ligne entière.

En fin de ligne, le commentaire commence par une apostrophe. Ex. :

```
Remise = Montant * 0.1    ' On calcule une remise de 10%
```

Sur ligne entière, le commentaire commence par une apostrophe ou le mot-clé `Rem`. On utilise plutôt l'apostrophe. Si le commentaire occupe plusieurs lignes, chaque ligne doit avoir son apostrophe.

Les lignes vides sont autorisées en VBA ; elles peuvent servir à aérer le texte. Nous conseillons de mettre une apostrophe en tête pour montrer que le fait que la ligne soit vide est voulu par le programmeur.

LES ESPACES

Les espaces sont assez libres en VBA, mais pas totalement. Là où il peut et doit y avoir un espace, vous pouvez en mettre plusieurs, ou mettre une tabulation.

On ne doit en aucun cas incorporer d'espaces à l'intérieur d'un mot-clé, d'un nom d'objet prédéfini, d'un nombre ou d'un nom de variable : ces mots ne seraient pas reconnus.

Au contraire, pour former des mots, ces éléments doivent être entourés d'espaces, ou d'autres caractères séparateurs comme la virgule.

Les opérateurs doivent être entourés d'espaces, mais vous n'êtes pas obligés de les taper, l'éditeur VBA les fournira sauf pour &. Si vous tapez `a=b+c` vous obtiendrez `a = b + c`.

LES RETRAITS OU INDENTATIONS

Les instructions faisant partie d'une même séquence doivent normalement commencer au même niveau d'écartement par rapport à la marge. Lors de l'emploi d'instructions de structuration, les séquences qui en dépendent doivent être en retrait par rapport aux mots-clés de structuration. En cas de structures imbriquées, les retraits doivent s'ajouter. Exemple fictif :

```
x = 3
For I = 2 To 10
    a = 0.05 * I
    If b < x Then
         x = x - a
    Else
         b = b – a
    End If
Next I
```

En cas de nombreuses imbrications, le retrait peut être un peu grand : bornez-vous à 2 caractères à chaque niveau. Bien sûr, ces retraits ne sont pas demandés par le langage, ils n'ont que le but de faciliter la compréhension en faisant ressortir la structure du programme (ou plutôt, la structure souhaitée, car, dans son interprétation, VBA ne tient compte que des mots-clés, pas des indentations : mais justement un désaccord entre les mots-clés et les indentations peut vous aider à dépister une erreur).

Il est donc essentiel, bien que non obligatoire que vous respectiez les indentations que nous suggèrerons pour les instructions.

AIDE À LA RECHERCHE D'ERREURS

Nous avons vu plus haut que VBA introduisait de lui-même les majuscules voulues dans les mots-clés et les noms de variables, d'où notre conseil de tout taper en minuscules : s'il n'y a pas de transformation, c'est qu'il y a probablement une faute de frappe.

Pour les mots-clés, on a une aide supplémentaire : VBA met les mots-clés en bleu (en fait, la couleur choisie par option) ; si un mot n'est pas transformé, c'est qu'il n'est pas reconnu, donc qu'il y a une faute.

Une autre aide automatique est que, en cas d'erreur de syntaxe, VBA affiche aussitôt un message d'erreur et met l'instruction en rouge. Bien sûr cela ne décèle que les erreurs de syntaxe, pas les erreurs de logique du programme.

AIDES À L'ÉCRITURE

L'éditeur VBA complète automatiquement certaines instructions :

Dès que vous avez tapé une instruction `Sub` ou `Function`, VBA fournit le `End Sub` ou le `End Function`.

Si vous tapez `endif` sans espace, VBA corrige : `End If`. Attention, il ne le fait que pour celle-là : pour `End Select` ou pour `Exit Sub` ou d'autres, il faut taper l'espace.

Dès que vous tapez un espace après l'appel d'une procédure, ou la parenthèse ouvrante à l'appel d'une fonction, VBA vous suggère la liste des arguments. Il le fait toujours pour un élément prédéfini ; pour une procédure ou fonction définie par vous, il faut qu'elle ait été définie avant.

Dès que vous tapez le `As` dans une déclaration, VBA fournit une liste déroulante des types possibles ; il suffit de double-cliquer sur celui que vous voulez pour l'introduire dans votre instruction. Vous avancez rapidement dans la liste en tapant la première lettre souhaitée. Un avantage supplémentaire est qu'un élément ainsi écrit par VBA ne risque pas d'avoir de faute d'orthographe.

De même, dès que vous tapez le point après une désignation d'objet, VBA affiche la liste déroulante des sous-objets, propriétés et méthodes qui en dépendent et vous choisissez comme précédemment. L'intérêt est que la liste suggérée est exhaustive et peut donc vous faire penser à un élément que vous aviez oublié. Attention, cela n'apparaît que si l'aide en ligne est installée et si le type d'objet est connu complètement à l'écriture, donc pas pour une variable objet qui aurait été déclarée d'un type plus général que l'objet désigné (ex. `As Object`).

PROJETS, DIFFÉRENTES SORTES DE MODULES

DÉFINITION

Un **projet** est l'ensemble de ce qui forme la solution d'un problème (nous ne voulons pas dire « application » car ce terme a un autre sens, à savoir l'objet Application, c'est-à-dire Excel lui-même), donc un classeur Excel avec ses feuilles de calcul, et tous les programmes écrits en VBA qui sont sauvegardés avec le classeur. Les programmes sont dans des modules ; le texte des programmes est affiché dans des fenêtres de code. Il peut y avoir un module associé à chaque feuille ou au classeur. Il peut y avoir un certain nombre de modules généraux. De plus, le projet peut contenir aussi des modules de classe et des boîtes de dialogue créées par le programmeur : chaque BDi a en principe un module de code associé.

Un programme peut ouvrir d'autres classeurs que celui qui le contient ; ces classeurs forment autant de projets, mais secondaires par rapport au projet maître.

LES FENÊTRES DU PROJET

L'écran VBA contient principalement la fenêtre de projet où apparaît le projet associé à chaque classeur ouvert. Chaque projet y apparaît sous forme d'une arborescence (développable ou repliable) montrant tous les éléments du projet. Sous la fenêtre de projet, peut apparaître une fenêtre Propriétés qui affiche les propriétés d'un élément choisi dans la fenêtre de projet ou d'un contrôle sélectionné dans une BDi en construction.

La plus grande partie de l'écran sera consacrée aux fenêtres de BDi en construction ou de code. Comme ces fenêtres sont en principe présentées en cascade, on choisit celle qui est en premier plan par clic dans le menu *Fenêtre*. On décide de l'affichage d'un tel élément par double-clic dans l'arborescence.

On peut faire apparaître d'autres fenêtres par clic dans le menu *Affichage*. C'est le cas des fenêtres de (l'Explorateur de) Projets, Propriétés, Explorateur d'objets, Exécution, Variables locales et Espions, ces trois dernières servant surtout au dépannage des programmes.

Le menu *Affichage* permet de basculer entre l'affichage d'un objet (comme une BDi) et la fenêtre de code correspondante (raccourci touche F7).

Le choix des fenêtres à afficher peut se faire aussi par des boutons de la barre d'outils Standard de l'écran VBA.

DIFFÉRENTES SORTES DE MODULES

À chacune des quatre rubriques de la hiérarchie dépendant du projet correspond une sorte de module. À *Microsoft Excel Objects* (les feuilles et le classeur) correspondent des modules où se trouveront les programmes de réponse aux événements de la feuille (ex. *Worksheet_Change*) ou du classeur (ex. *Workbook_Open*).

À *Feuilles* correspondent les BDi construites par le programmeur (UserForms). Chacune a un module associé qui contient les procédures de traitement des événements liés aux contrôles de la BDi (ex. *UserForm_Initialize*, *CommandButton1_Click,* etc.) ;

À *Modules* correspondent les différents modules « normaux » introduits. C'est dans ces modules (en principe, on les regroupe en un seul) que sont les procédures de calcul propres au problème.

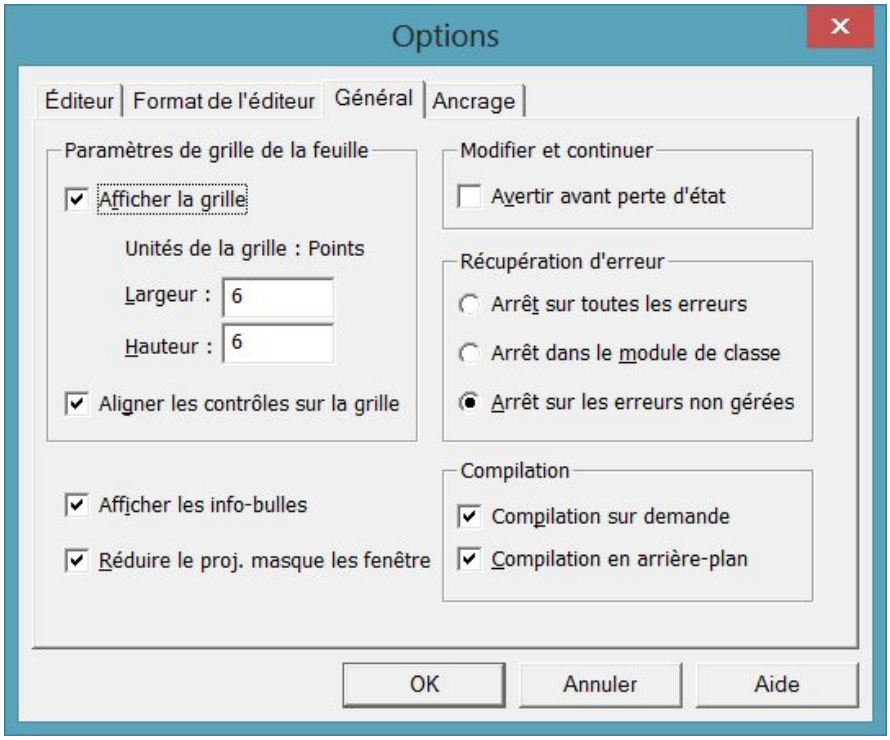
La dernière sorte de modules dépend de la rubrique *Modules de classe* ; les modules de classe permettent de définir des objets propres au programmeur. Ils sont beaucoup moins souvent utilisés car, vu la richesse des objets prédéfinis en Excel VBA, on en utilise rarement plus de 10 %, alors on a d'autant moins de raisons d'en créer d'autres !

Une dernière rubrique, *Références* peut être présente dans l'arborescence, mais elle n'introduit pas de modules.

OPTIONS DE PROJETS

LA COMMANDE OUTILS-OPTIONS

Cette commande concerne les projets par ses onglets *Général* et *Ancrage*. L'onglet *Ancrage* décide quelles fenêtres vont pouvoir être ancrées c'est-à-dire fixées en périphérie de l'écran. Ce n'est pas vital. L'onglet *Général* a plus à dire :

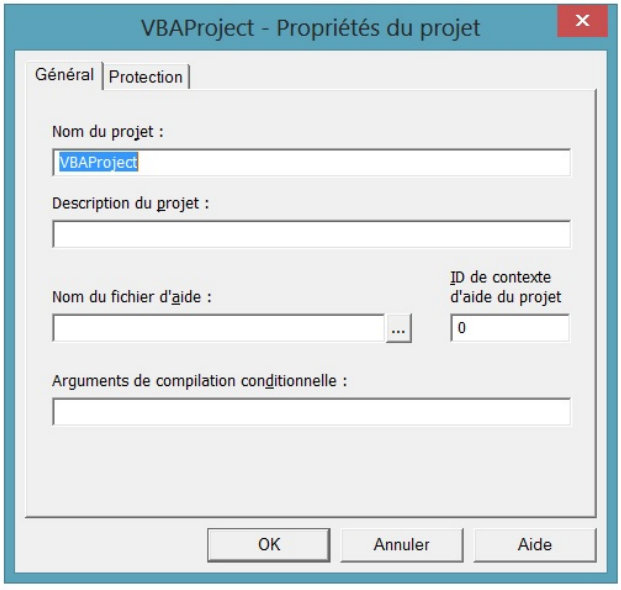


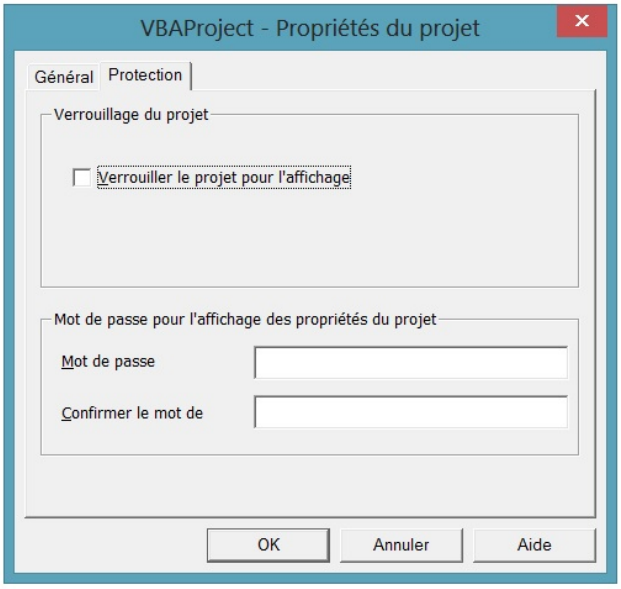
- Le cadre Paramètres de grille de la feuille gère le placement des contrôles sur une BDi construite par le programmeur, donc voir chapitre 6.
- ☑ *Afficher les info-bulles* affiche les infobulles des boutons de barre d'outils.
- ☑ *Réduire le proj. masque les fenêtres* définit si les fenêtres de projet, UserForm, d'objet ou de module sont fermées automatiquement lors de la réduction du projet dans l'Explorateur de projet.
- Le cadre Modifier et continuer.
  - ☑ *Avertir avant perte d'état* active l'affichage d'un message lorsque l'action demandée va entraîner la réinitialisation de toutes les variables de niveau module dans le projet en cours.
- Le cadre Récupération d'erreur définit la gestion des erreurs dans l'environnement de développement Visual Basic. L'option s'applique à toutes les occurrences de Visual Basic lancées ultérieurement.
  - ⊙ *Arrêt sur toutes les erreurs* : en cas d'erreur quelle qu'elle soit, le projet passe en mode Arrêt.
  - ⊙ *Arrêt dans les modules de classe* : en cas d'erreur non gérée survenue dans un module de classe, le projet passe en mode Arrêt à la ligne de code du module de classe où s'est produite l'erreur.
  - ⊙ *Arrêt sur les erreurs non gérées* : si un gestionnaire d'erreurs est actif, l'erreur est interceptée sans passage en mode Arrêt. Si aucun gestionnaire d'erreurs n'est actif, le projet passe en mode Arrêt. Ceci est l'option la plus conseillée.

- Compilation
  - ☑ *Compilation sur demande* définit si un projet est entièrement compilé avant d'être exécuté ou si le code est compilé en fonction des besoins, ce qui permet à l'application de démarrer plus rapidement, mais retarde l'apparition des messages d'erreur éventuels dans une partie de programme rarement utilisée.
  - ☑ *Compilation en arrière-plan* définit si les périodes d'inactivité sont mises à profit durant l'exécution pour terminer la compilation du projet en arrière-plan, ce qui permet un gain de temps. Possible seulement en mode compilation sur demande.

LA COMMANDE OUTILS-PROPRIÉTÉS DE <NOM DU PROJET>

Cette commande fait apparaître une BDi avec deux onglets :





- L'onglet *Général* permet de donner un nom plus spécifique que VBAProject, et surtout de fournir un petit texte descriptif. Les données concernant l'aide n'ont plus d'intérêt : la mode est maintenant de fournir une aide sous forme HTML. La compilation conditionnelle est sans réel intérêt.
- L'onglet *Protection* permet de protéger votre travail.
  - ☑ *Verrouiller le projet pour l'affichage* interdit toute modification de n'importe quel élément de votre projet. Il ne faut y faire appel que lorsque le projet est parfaitement au point !
  - La fourniture d'un mot de passe (il faut le donner deux fois, c'est classique) empêche de développer l'arborescence du projet dans la fenêtre Explorateur de projets si l'on ne donne pas le mot de passe. Donc un « indiscret » qui n'a pas le mot de passe n'a accès à aucune composante de votre projet.

LA COMMANDE OUTILS-RÉFÉRENCES

Permet de définir une référence à la bibliothèque d'objets d'une autre application pour y sélectionner des objets appartenant à cette application, afin de les utiliser dans votre code. C'est une façon d'enrichir votre projet.

LES DIFFÉRENTES SORTES D'INSTRUCTIONS

Les instructions VBA se répartissent en instructions exécutables ou ordres et instructions non exécutables ou déclarations.

INSTRUCTIONS EXÉCUTABLES

Ce sont les instructions qui font effectuer une action par l'ordinateur. Elles se répartissent en :

- **Instructions séquentielles**, telles que l'instruction qui sera exécutée après est l'instruction qui suit dans le texte.
  - La principale instruction de cette catégorie est *l'instruction d'affectation*, de la forme `[Set]<donnée>=<expression>`, où l'expression indique un calcul à faire. L'expression est calculée et le résultat est affecté à la donnée. En l'absence de `Set` (on devrait normalement mettre `Let`, mais il n'est jamais employé), l'expression conduit à une valeur et `<donnée>` est une variable ou une propriété d'objet ; elle reçoit la valeur calculée comme nouvelle valeur. Avec `Set`, l'expression a pour résultat un objet et `<donnée>` est une variable du type de cet objet : après l'instruction, cette variable permettra de désigner l'objet de façon abrégée. À part l'appel de procédures, cette instruction est la plus importante de tout le langage.
  - *Toute une série d'actions diverses*, notamment sur les fichiers (`Open`, `Close`, `Print#`...) ou sur certains objets (`Load`, `Unload`...) ou encore certaines opérations système (`Beep`, `Time`…). Ces instructions pourraient d'ailleurs aussi bien être considérées comme des appels à des procédures ou des méthodes prédéfinies.

- **Instructions de structuration**, ou de rupture de séquence, qui rompent la suite purement linéaire des instructions, aiguillant le traitement vers une séquence ou une autre selon des conditions, ou faisant répéter une séquence selon les besoins. Ces instructions construisent donc la structure du programme. La plus importante est :
  - *L'appel de procédure* : on déroute l'exécution vers un bloc d'instructions nommé qui remplit un rôle déterminé. La fin de l'exécution de la procédure se réduit à un retour dans la procédure appelante juste après l'instruction d'appel. Cela permet de subdiviser un programme complexe en plusieurs petites unités beaucoup plus faciles à maîtriser. La plupart du temps, l'instruction se réduit à citer le nom de la procédure à appeler.

Les autres instructions de structuration permettent d'implémenter les deux structures de la programmation structurée.

- *La structure alternative* où, en fonction de certaines conditions, on fera une séquence ou bien une autre. VBA offre pour cela deux instructions principales, `If` qui construit une alternative à deux branches et `Select Case` qui permet plusieurs branches.
- *La structure itérative* ou boucle, où on répète une séquence jusqu'à ce qu'une condition soit remplie (ou tant que la condition contraire prévaut). VBA offre pour cette structure les instructions `Do…Loop`..., `While…Wend` et, surtout, `For…Next` qui est la plus employée.

INSTRUCTIONS NON EXÉCUTABLES OU DÉCLARATIONS

Ces instructions ne déclenchent pas d'actions de l'ordinateur, mais donnent des précisions au système VBA sur la manière dont il doit traiter les instructions exécutables. La plus importante de ces instructions est la déclaration de variable qui :

- Annonce qu'on va utiliser une variable de tel ou tel nom.
- Indique le type (par exemple réel, ou entier, etc.) de la variable, c'est-à-dire des données qu'elle va contenir. Il est évident que les calculs ne s'effectuent pas de la même façon sur un nombre entier ou sur un réel. C'est en cela que les déclarations orientent le travail de VBA. **Elles sont donc aussi importantes que les instructions exécutables**.

Place des déclarations de variables

Normalement, il suffit qu'une déclaration de variable soit n'importe où avant la première utilisation de cette variable. En fait on recommande vivement de placer les déclarations de variables en tête de leur procédure. Par ailleurs, certaines déclarations de variables doivent être placées en tête de module, avant la première procédure du module.

Parmi les déclarations importantes, les couples `Sub … End Sub` et `Function … End Function` délimitent respectivement une procédure ou une fonction. `Sub` et `Function` ont en outre le rôle de déclarer des éventuels arguments. Les deux `End …` sont à la fois des déclarations – elles délimitent la fin de la procédure ou de la fonction – et des instructions exécutables : lorsque l'on arrive sur elles on termine la procédure ou la fonction et on retourne à l'appelant.

DIRECTIVES

Les directives sont des déclarations particulières qui jouent un rôle global au niveau du projet. Elles sont placées tout à fait en tête de module. Certaines peuvent être spécifiées sous forme d'options de projet auquel cas la directive est écrite automatiquement en tête de tous les modules.

`Option Explicit`

Exige que toute variable soit déclarée. Nous conseillons vivement cette option car si vous faites une faute de frappe dans un nom de variable, en l'absence de cette option, VBA « croira » que vous introduisez une nouvelle variable, alors qu'avec cette option, il y aura un message d'erreur vous permettant de la corriger aussitôt.

`Option Base <0 ou 1>`

Fixe à 0 ou à 1 la première valeur des indices de tableaux. La valeur par défaut est 0. Souvent les programmeurs utilisent les indices à partir de 1 sans spécifier `Option Base 1` : l'élément 0 est laissé vide. Cette pratique a un inconvénient : si par erreur un indice était calculé à 0, la directive assurerait un message d'erreur.

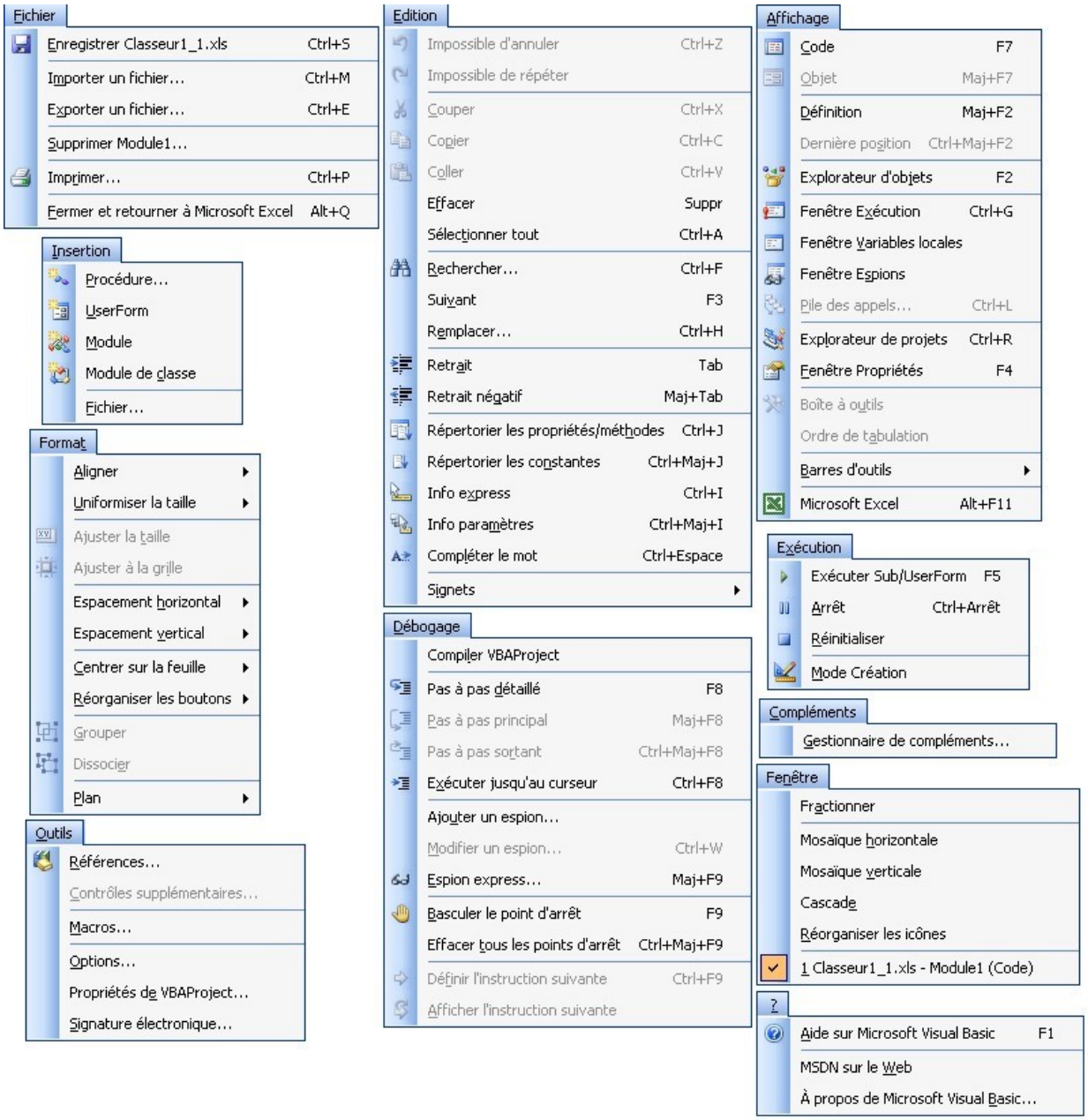
`Option Compare <choix>`

Fixe la façon dont les chaînes de caractères sont comparées. Avec `Text`, une majuscule et sa minuscule sont confondues alors qu'avec `Binary`, la comparaison est complète et les minuscules sont plus loin que les majuscules dans l'ordre alphabétique.

`Option Private Module`

Déclare le module entier comme privé, donc aucun de ses éléments, variables, procédures ou fonctions ne sera accessible depuis un autre module.

LES MENUS DE L'ÉDITEUR VBA



N.B. Certaines rubriques peuvent varier légèrement en fonction du contexte, selon qu'on est dans une procédure ou non et selon ce qu'on a fait précédemment ; ainsi *Edition – Impossible d'annuler* peut devenir *Edition – Annuler*, *Exécuter Sub...* peut devenir *Exécuter la macro,* etc.