Université Libre de Bruxelles

Section des sciences économiques et Solvay Business School
Année académique 2004-2005

GUIDE D'APPRENTISSAGE DES MACROS d'EXCEL

par **Guy Mélard, Marc Colet et Hassane Njimi (avec la collaboration de Uwe Prasser)**
© 2ère édition, 21 mars 2005
basé sur "Guide d'apprentissage des fonctions et des macros d'Excel"
1ère édition, 10 mars 2004 par Guy Mélard, Atika Cohen et Marc Colet

PREFACE

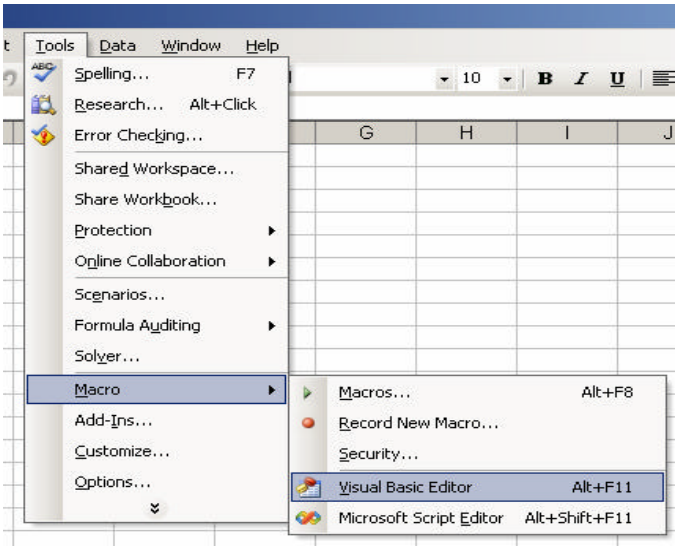
Ce texte constitue une introduction à la macro-programmation en Excel 2003. Disponible depuis la version 5.0 d'Excel, le langage de programmation s'appelle **V**isual **B**asic for **A**pplications (le langage de programmation des applications de Microsoft Office, en abrégé VBA). Nous conseillons aux lecteurs intéressés de consulter un ouvrage tel que "Visual Basic pour Applications 6", de Mikaël Bidault, Editions CampusPress, Paris, 1999. Nous nous limitons à quelques possibilités offertes par les macros afin d'illustrer l'algorithmique et la programmation.

Il faut noter que la macro-programmation VBA est disponible également sous Word, PowerPoint, Outlook et FrontPage (à partir de la version 6 sous Office2000). VBA permet d'automatiser les tâches, de créer des applications complètes, de sécuriser vos saisies et vos documents, de créer de nouveaux menus et de nouvelles fonctions pour améliorer efficacement votre logiciel.
VBA, langage puissant, souple et facile à utiliser permet de réaliser très rapidement des applications qui vous feront économiser du temps et de l'argent. Les applications VBA pour Excel sont abritées dans un document Excel ou classeur (appelé "workbook") mais peuvent manipuler des informations situées dans d'autres documents de Microsoft Office. Plus généralement, VBA doit fonctionner dans un environnement approprié. Par exemple un projet VBA pour Excel nécessite l'ouverture dans Excel d'un classeur Excel contenant les modules du programme. Une application VBA créée sous Excel ne pourra pas se lancer sur un poste si Excel n'est pas installé.

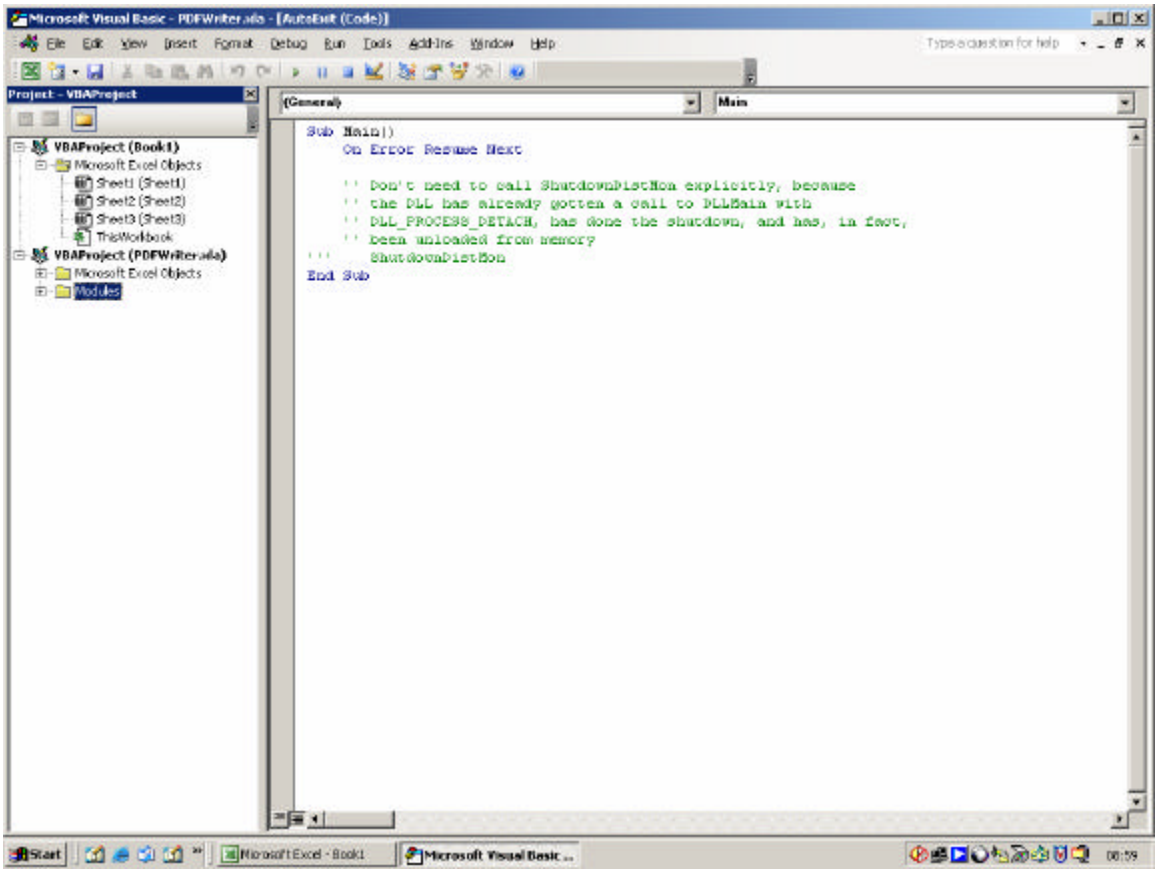
Il existe également un langage de programmation appelé Visual Basic tout court, maintenant une partie de Visual Studio .NET, qui permet de développer sous Microsoft Windows (Windows NT/2000/XP) des applications qui ne sont pas liées à Microsoft Office.
Le Visual Basic et le Visual Basic for Applications sont dérivés du langage Basic qui existe depuis 1965 (Basic est l'acronyme de Beginner's All-purpose Symbolic Instruction Code). Il a été conçu par Tom Kurtz et John Kemeny du Dartmouth College. Basic (sous les noms Basica ou GWBasic) était le langage de programmation au premier temps des PC. Il a évolué en passant par le Qbasic (inclus sous MS-DOS depuis la version 5.0). Les macro-programmes d'Excel 4.0 n'étaient pas basés sur Basic mais fonctionnent toujours dans les versions ultérieures d'Excel. Le langage Basic de VBA est compatible avec le Liberty Basic (illustré dans "Débuter en Programmation", de Greg Perry, Editions CampusPress, Paris, 2002) et avec le S-Basic de la suite libre OpenOffice.org et de StarOffice de Sun. Toutefois, les objets (fenêtres, boutons et autres contrôles) ainsi que leurs méthodes et leurs propriétés ne sont pas compatibles.

CHAPITRE 1 VBA : L'EDITEUR DE MACRO

Lancez Excel. L'éditeur de macro, ou VBE (Visual Basic Editor) est l'environnement de programmation de VBA. Il se lance par le menu "Outils-Macro-Visual-Basic-Editor" ou par le raccourci clavier "Alt+F11".



1.1 Les principales fenêtres de VBE :



1- Fenêtre VBAProject. Elle présente les différents projets ouverts et permet de naviguer facilement entre vos différentes feuilles de codes VBA.

2 - Fenêtre Code. C'est l'endroit ou vous allez saisir votre code VBA.

3 - Fenêtre Propriétés. Propriétés de l'objet sélectionné.

4- Fenêtre Exécution. Elle permet de tester une partie du code. Elle peut s'avérer très utile pour voir comment s'exécutent certaines lignes de code.

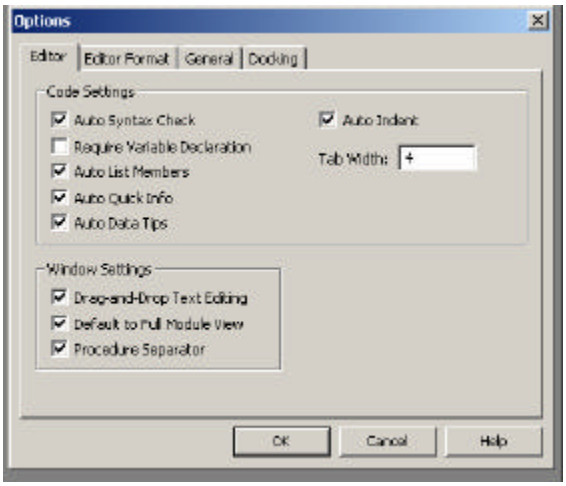
Il est fort probable que l'aspect de votre éditeur de macros soit différent. Il est en effet personnalisable car chaque fenêtre peut être masquée puis réaffichée par le menu "Affichage". Cependant, cette configuration vous permettra de débuter de façon confortable l'écriture de vos premières macros.

1.2 Configuration de l'éditeur de macros :

Il est important de bien configurer l'éditeur de macros. En effet, VBE peut vous aider dans l'écriture de votre code et le mettre en forme de façon à ce qu'il soit plus facile à lire.

Sous VBE, lancer le menu "Outils-Options" :

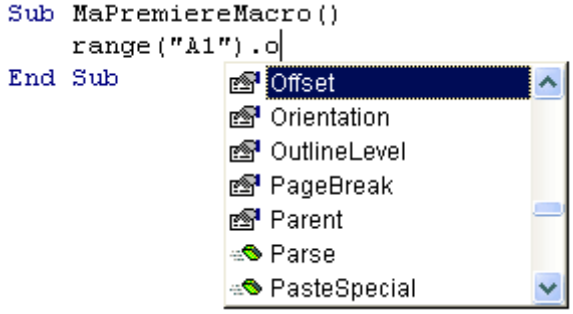
1 - Onglet Editeur :



- Vérification automatique de la syntaxe : vérification automatiquement de la syntaxe lors de la saisie d' une ligne de code.

Déclarations de variables obligatoires : sous VBA, la déclaration de variables n'est pas obligatoire. Cependant, je vous conseille de cocher cette option. De plus amples informations au sujet des variables seront disponibles dans le cours "Les variables". Si la case est cochée, l'instruction "Option Explicit" est ajoutée dans les déclarations générales de tout nouveau module.

- Complément automatique des instructions : cette option permet à VBE de vous aider dans la saisie de votre code.



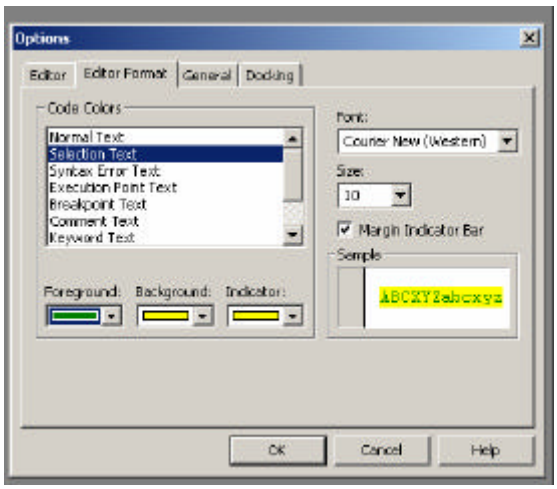
Vous comprendrez très vite son utilité lorsque vous saisirez vos premières lignes de codes.

- Info express automatique : encore une option très utile. Elle affiche les différents arguments que possède la fonction que vous venez de taper.

```
Sub MaPremiereMacro()
    range(
End  Range(Cell1, [Cell2]) As Range
```

- Info-bulles automatique : indispensable lors d'un déboguage pas à pas. Elle permet l'affichage de vos variables.
- Retrait automatique : permet à VBE de placer chaque ligne de code au même niveau que la ligne précédente. Le retrait de lignes se fait par les touches "Tab" et "Shift+Tab". Cette option est nécessaire pour une bonne lecture du code VBA.
- Paramètres de la fenêtre : les 3 options sont intéressantes. L'édition de texte par glisser-déplacer permet de déplacer à l'aide de la souris le bloc de code sélectionné, l'affichage du module complet par défaut permet l'affichage de toutes les procédures d'un même module et la séparation des procédures oblige VBE à créer des traits entre chaque procédure.

2 - Onglet Format de l'éditeur :



Cet onglet permet de changer la police et son format pour les différentes parties du code inscrit dans vos modules. On ne vous conseille pas de changer les paramètres par défaut.

Ces différentes options vous permettent de configurer à votre convenance l'éditeur de macros. Si vous débutez, il est conseillé de garder les options par défaut.

CHAPITRE 2 A PROPOS DES MACROS

Les fonctions sont des outils de calcul que vous pouvez utiliser pour aider à la prise de décision, à l'exécution d'actions et à l'obtention automatique de synthèses de données. Excel propose deux types de fonctions: les fonctions de feuille de calcul (que vous avez déjà utilisées) et les fonctions macro. Vous pouvez utiliser les fonctions de feuille de calcul à la fois dans les feuilles de calcul et

dans les macros. Par exemple, la fonction **SUM** (utilisée dans le texte sur Excel) calcule la somme de plusieurs nombres; il s'agit d'une fonction feuille de calcul également disponible dans une macro. Certaines fonctions ne sont disponibles que dans les feuilles macro. De nombreuses fonctions macro acceptent et renvoient également des valeurs.

Les macros sont des mini-programmes que l'on édite dans des feuilles macro. Plusieurs macros peuvent être éditées dans une feuille. Signalons enfin que les opérateurs (+, - , / *, &, ^, ...) sont disponibles aussi bien dans les feuilles de calcul que dans les macros.

Enregistrement des macros

La manière la plus simple de créer une macro dans Excel est de l'enregistrer. Cela permet déjà d'automatiser des tâches et donc de les accélérer. L'enregistrement peut comporter des déplacements dans la feuille, des actions correspondant aux commandes des menus et à l'état final des boîtes de dialogues qui sont activées, y compris le recours aux fonctions et aux modules supplémentaires. Néanmoins il n'est pas possible de tout faire de cette manière. Ni les interactions (affichages de message, saisie de données à l'exécution), ni le contrôle du programme (instructions pour réaliser des conditions et des boucles) ne sont susceptibles d'être enregistrées parce qu'elles ne correspondent pas à des commandes des menus. Pourtant, on recommande de commencer par enregistrer des séquences d'opérations (éventuellement de manière séparée), de modifier si nécessaire le programme VBA produit et de les relier ensuite dans l'application, ou mieux encore dans des procédures isolées qui seront ensuite assemblées pour réaliser l'application.

Lors de l'enregistrement d'une macro, il est essentiel de choisir entre adressage absolu et adressage relatif. En adressage absolu, la ligne et la colonne des cellules sont enregistrées à chaque sélection. En adressage relatif, ce sont seulement les déplacements en nombre de lignes et de colonnes, en plus (à droite ou vers le bas) ou en moins (à gauche ou vers le haut) qui sont enregistrés. A tout moment on peut passer de l'un à l'autre des adressages en cliquant sur le bouton de droite de l'enregistreur de macro.

Ecriture des macros

Comme nous l'avons dit, la syntaxe des macros d'Excel s'appuie sur le langage Basic. Dans ce texte, nous nous référons au langage de description d'algorithme (LDA) (étudié notamment dans le cours "Modélisation partie informatique" de G. Mélard, disponible aux Presses Universitaires de Bruxelles). Une comparaison détaillée est donnée dans le chapitre 3 qui n'est pas essentiel en première lecture.

Le lignes de commentaires commencent par une apostrophe (‘). Les lignes trop longues peuvent être coupées en terminant chaque ligne sauf la dernière par un caractère souligné "_". On doit veiller à ne pas couper un nom ou une constante en deux parties. La syntaxe d'une ligne de commande est la suivante:

```
variable = expression
```

où la variable est parfois indispensable, parfois interdite ou encore facultative. Dans sa version la plus simple, une expression peut être une constante (0, par exemple), une variable (*var*, par exemple) ou une fonction d'Excel, par exemple:

```
FonctionXl( argument_1, argument_2, argument_3)
```

Si la variable est omise, on peut employer une procédure, de la forme

```
ProcedureXl  argument_1  argument_2  argument_3
```

Le nombre d'arguments dépend de la fonction ou de la procédure. Certains arguments peuvent être omis lors d'un appel de fonction. Ils peuvent alors être indiqués par "," suivi de ",".

Excel vous informe du nombre et du type d'arguments désirés. D'autre part, l'interpréteur des macros Excel contrôle le nombre d'arguments ainsi que leur type lors de l'édition de la macro.

Exemples:

1. La fonction **MsgBox** affiche une boîte de dialogue pour les entrées de l'utilisateur. C'est une fonction uniquement réservées aux macros:

***variable* = MsgBox(*texte_message*, *type_boutons*, *texte_titre*)**

Trois arguments sont offerts mais un seul est obligatoire:

- ***texte_message*** : message affiché dans la boîte de dialogue (de type String);
- ***type_boutons*** : facultatif, nombre qui indique le type de boutons dans la boîte de dialogue
  0 ou vbOKOnly: bouton OK,
  1 ou vbOKCancel: boutons OK et Cancel,
  3 ou vbYesNoCancel: boutons Yes , No et Cancel,
  4 ou vbYesNo: boutons Yes et No,
  ajouter 16 ou vbCritical, pour avoir en plus un symbole "X",
  ajouter 32 ou vbQuestion, pour avoir "?",
  ajouter 48 ou vbExclamation pour avoir "!",
  ajouter 64 ou vbInformation pour avoir "i" ;
- ***texte_titre*** : facultatif, titre de la boîte de dialogue (de type String).

La valeur de ***variable*** dépend du bouton enfoncé par l'utilisateur:
1 (ou vbOK) si c'est OK,
2 (ou vbCancel) si c'est Cancel,
6 (ou vbYes) si c'est Yes,
7 (ou vbNo) si c'est No.
On peut employer une instruction conditionnelle (voir exemple 3 ci-dessous).

La présence de ***variable*** n'est pas toujours indispensable, notamment pour afficher un message. Si on l'omet (ainsi que le signe "="), il faut employer la variante procédure de la fonction MsgBox qui s'écrit de la même manière mais sans les parenthèses:

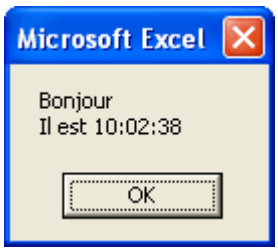
MsgBox *texte_message type_boutons texte_titre*

NB : les **MsgBox** peuvent simplement donner une information. La procédure est alors stoppée tant que l'utilisateur n'a pas cliqué sur le bouton.

MsgBox "Bonjour"

Le texte peut-être affiché sur plusieurs lignes en utilisant le code retour chariot chr(13) ou le code retour ligne chr(10).

MsgBox "Bonjour" & Chr(10) & "Il est " & Time



2. La fonction **InputBox** affiche une boîte de dialogue contenant une zone de texte pour inviter l'utilisateur à y saisir une information.

variable **= InputBox(*texte_message, texte_titre, défaut*)**

Trois arguments sont offerts mais un seul est obligatoire:

- ***texte_message*** : message affiché dans la boîte de dialogue (de type String);
- ***texte_titre*** : facultatif, titre de la boîte de dialogue (de type String),
- ***défaut*** : facultatif, valeur par défaut qui apparaît dans la zone de texte au moment de l'affichage.

La valeur de ***variable*** dépend de la zone de texte saisie et est de type String. Elle sera éventuellement convertie en nombre lors du traitement. Si l'utilisateur clique sur le bouton Annuler, une erreur se produit. Elle doit être capturée par une clause "On Error" du Visual Basic.

La présence de la variable est ici indispensable. La variable prendra la valeur saisie au clavier. Par exemple:

```
nom = InputBox("Entrer votre nom", "Fenêtre d'encodage", "Votre nom")
```

La variable nom est de type String. Elle est fixée à "Votre nom" par défaut et prendra la valeur saisie au clavier. On peut aussi employer

```
Age = InputBox("Entrer votre âge", , 20)
```

Contrairement à la documentation, la valeur de la variable Age peut être traitée de manière numérique. Il n'est pas nécessaire d'employer une instruction AgeNum = Val(Age) pour la conversion numérique. La longueur maximum de ***texte_message*** est de 1024 caractères. Pour les écrire à l'écran sur plusieurs lignes on peut séparer plusieurs chaînes avec le retour chariot (Chr(13).

3. La structure conditionnelle (plus puissante que la fonction If interne d'Excel) se présente comme suit :

```
If condition Then
      bloc d'instructions
Else
      bloc d'instructions
End If
```

On peut compléter cette instruction de contrôle par une ou plusieurs clauses **ElseIf**. Exemple

```
If Age < 25 Then
      MsgBox "Vous avez droit à une réduction pour jeunes"
ElseIf Age >= 65 Then
      MsgBox "Vous avez droit à une réduction troisième âge"
Else
      MsgBox "Vous payez le prix plein"
End If
```

On peut aussi omettre Else et le bloc d'instruction qui suit. On peut aussi utiliser la syntaxe simplifiée et supprimer Then et End If mais alors le bloc d'instructions doit soit être composé d'une seule instruction, soit être composé de plusieurs instructions sur une seule ligne avec des ":" comme séparateurs.

CHAPITRE 3 CREATION ET EXECUTION DE MACROS

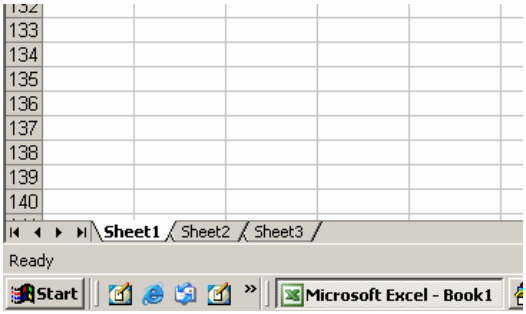
3.1 INTRODUCTION

VBA manipule les objets de l'application hôte. Chaque objet possède des propriétés et des méthodes.

Les objets :

Chaque objet représente un élément de l'application. Sous Excel, un classeur, une feuille de calcul, une cellule, un bouton, etc ... sont des objets. Par exemple, Excel représente l'objet Application, Workbook l'objet classeur, Worksheet l'objet feuille de calcul etc...

Tous les objets de même type forment une collection comme, par exemple, toutes les feuilles de calcul d'un classeur. Chaque élément est alors identifié par son nom ou par un index.



Pour faire référence à la deuxième feuille (ici Sheet2), on va utiliser Worksheets(2) ou Worksheets("Sheet2")

Chaque objet peut avoir ses propres objets. Par exemple, Excel possède des classeurs qui possèdent des feuilles qui possèdent des cellules. Pour faire référence à une cellule, on pourrait ainsi utiliser :

Application.Workbooks(1).Worksheets("Sheet2").Range("A1")

Les propriétés :

Une propriété correspond à une particularité de l'objet. La valeur d'une cellule, sa couleur, sa taille, ..., sont des propriétés de l'objet Range. Les objets sont séparés de leurs propriétés par un point. On écrira ainsi Cellule.Propriété = valeur, par exemple pour mettre la valeur 10 dans la cellule A1 :

Range("A1").Value = 10

Une propriété peut également faire référence à un état de l'objet. Par exemple, si on veut masquer la feuille de calcul "Sheet2", on écrira :

Worksheets("Sheet2").Visible = False

Les méthodes :

On peut considérer qu'une méthode est une opération que réalise un objet. Les méthodes peuvent être considérées comme des verbes tels que ouvrir, fermer, sélectionner, enregistrer, imprimer, effacer, etc... Les objets sont séparés de leurs méthodes par un point. Par exemple, pour sélectionner la feuille de calcul nommé "Feuil2", on écrira :

Worksheets("Sheet2").Select

Lorsque l'on fait appel à plusieurs propriétés ou méthodes d'un même objet, on fera appel au bloc d'instruction **With** Objet *Instructions* **End With.** Cette instruction rend le code souvent plus facile à lire et plus rapide a exécuter.

'Mettre la valeur 10 dans la cellule A1, la police en gras et en italique et copier la cellule.

```
With Worksheets("Sheet2").Range("A1")
.Value = 10
.Font.Bold = True
.Font.Italic = True
.Copy
End With
```

Ce vocabulaire peut paraître déroutant mais deviendra très rapidement familier lors de la création de vos premières applications.

Avant de commencer :

Attention aux virus !

Bien qu'ils soient rares, il existe des virus qui s'attaquent à des documents Excel. La plupart sont des virus de type macro-commande. C'est pour cette raison qu'Excel vous demande lorsqu'il ouvre un document ayant une macro-commande s'il doit les activer ou non. Si vous savez que vous avez créé des macros pour vos besoins, vous pouvez répondre " Autoriser les macros : **Enable** ". Sinon, ne courez pas de risque et répondez " Désactiver les macros : **Disable** ".

Les virus de macro commandes étaient si "populaire" à une période que Microsoft a réagi en ajoutant un niveau de sécurité sur les macro aux versions d'Excel 2000 et les suivantes.

Du menu **Tools**, sélectionnez les options **Macro** et **Security**.

Vous pouvez choisir parmi trois niveaux de sécurité.

Niveau	**Description**
High	N'active pas les macros commandes inclus dans un fichier Excel.
Medium	Vous demande l'autorisation pour activer les macro-commandes incluses avant l'ouverture du fichier.
Low	Ouvre le fichier Excel sans aucune vérification des macro-commandes. Expose votre ordinateur aux risques reliés des macros commandes.

C'est votre décision de choisir le niveau de protection qui répond à vos besoins. On vous recommande de mettre le niveau de sécurité à **Medium**. Excel va vous demander l'autorisation d'ouvrir un fichier ayant des macros commandes. Il faut se protéger lorsqu'on considère le nombre de virus qui se propagent sur l'Internet. C'est un compromis entre la sécurité et une option pratique.

Du répertoire **F:\MELARD,** recopiez sur votre unité **H:** le fichier " EX03CH4.XLS". Ouvrez ce fichier et sauvez-le sous le nom **VB1G*g*B*bb*.xls**, où ***g*** est le numéro de série et ***bb***, le numéro de binôme. C'est l'exercice du chapitre 4 du Guide d'Apprentissage d'Excel, partiellement résolu.

Dans le champ D4:F6 de la feuille de travail « Emprunt_unique », on a inséré les données concernant trois modalités d'emprunt nommées Plan 1, Plan 2 et Plan3.

Nous allons créer quatre macros. Trois seront obtenues par enregistrement. La quatrième sera écrite. Les quatre macros porteront les noms suivants :

Emprunt_Initial	Il faudra recopier le champ D4:D6 sur le champ B4:B6.
Emprunt_à_5_ans	Il faudra recopier la cellule E6 dans la cellule B6.
Réduction_Montant_Emprunté	Il faudra recopier le champ F4:F6 sur le champ B4:B6.
Quantité_de_Limousine	Il s'agit de modifier, dans la feuille de travail « emprunts_multiples », la Quantité des limousines à acquérir figurant dans la cellule C7 pour parvenir à l'objectif de 44 passagers dans la cellule B15 avec un total des mensualités inférieur ou égal à 3700€ dans la cellule B14.

Revenez au classeur Excel.

3.2 ENREGISTREMENT D'UNE MACRO

Enregistrons la macro Emprunt_Initial. Dans le menu Tools, cliquez sur Macro, Record a new macro et complétez le nom dans la zone Macro Name. Une petite fenêtre de contrôle d'enregistrement s'ouvre. Le bouton avec le carré permettra d'arrêter l'enregistrement. Entre-temps, la plupart de vos actions dans Excel seront enregistrées.

Sélectionnez les cellules à copier de D4 à D6. Copiez la sélection. Collez les valeurs (menu Edit Paste Special Values) dans la cellule B4. Arrêtez l'enregistrement.

Vous pouvez vérifier la macro en l'exécutant. Dans le menu Tools, cliquez sur Macro, Macro et sélectionnez le nom Emprunt_Initial. Cliquez sur Run.

Par curiosité, vous pouvez voir le code généré comme suit. Dans le menu Tools, cliquez sur Macro, Macro et sélectionnez le nom Emprunt_Initial. Cliquez sur Edit. La fenêtre du Visual Basic s'ouvre. Certaines lignes pourraient être supprimées. Ne supprimez rien maintenant. Ne fermez pas le Visual Basic mais retournez à Excel (cliquez sur le bouton Excel sur la barre des tâches ou sur le bouton Excel dans la barre d'outils de Visual Basic.

3.3 ENREGISTREMENT DE DEUX AUTRES MACROS

Enregistrons la macro Emprunt_à_5_ans. Arrangez-vous pour que la cellule E6 soit copiée sur la cellule B6.

Vous pouvez vérifier la macro en l'exécutant. Dans le menu Tools, cliquez sur Macro, Macro et sélectionnez le nom Emprunt_à_5_ans. Cliquez sur Run. Consultez ensuite le code correspondant. Vous pouvez profiter de l'occasion pour étendre la copie sur les cellules E4:E5. Remplacez E6 par E4:E6. Sauvez. Retournez à Excel. Refaites la même chose pour enregistrer la macro Réduction_Montant_Emprunté.

3.4 CREATION D'UNE MACRO

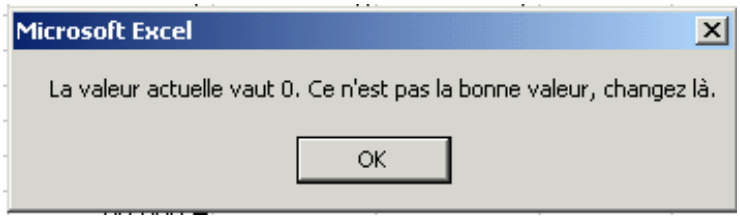
La cellule B17 contient une formule qui permet d'indiquer si la quantité des limousine à acquérir sous contraintes (voir détails des contraintes), est la bonne ou pas. Nous l'avons cachée (en employant un format de cellule «;;;»). Nous allons écrire une macro

- qui affiche le contenu de cette cellule en fonction de la valeur de la quantité des limousine à acquérir dans la cellule C7
- qui demande à l'utilisateur d'entrer une nouvelle valeur qui est placée dans la cellule C7.

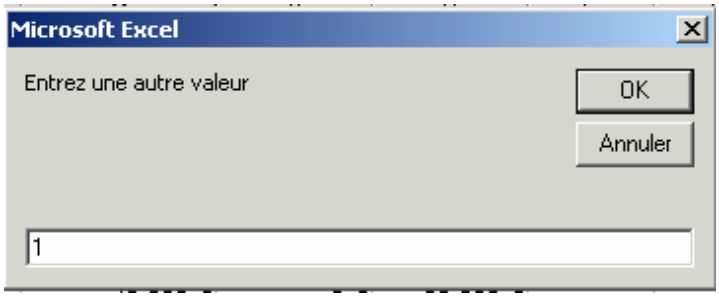
La première étape est réalisée à l'aide de la fonction MsgBox dans sa variante procédure. Un seul argument sera utilisé : le texte du message. Il sera obtenu par concaténation de mots et de contenus de cellules. Les mots doivent être placés entre guillemets. La concaténation emploie l'opérateur & entouré d'espaces. Les valeurs de cellules doivent être de la forme Range("B17").Value

Remarque : Rapellons que Range("B17").Value signifie ce qui permet d'accéder à la valeur contenue dans la cellule B17.

Ci-dessus un exemple de message à afficher.



Pour la saisie d'une nouvelle valeur, on doit obtenir la fenêtre suivante



Pour y parvenir, utilisez la fonction InputBox. Ceci se fait comme suit :

Range("C7").Value = InputBox("Entrez une autre valeur ")

Si vous prenez goût à Visual Basic et aux macros, vous pouvez affecter la macro que vous venez de réaliser à un bouton à définir dans votre feuille.

Pour y parvenir, choisir dans le menu View , activez la commande ToolBars, puis choisir Form. Sélectionnez Button puis suivre les instructions où l'on vous demande de préciser à quel macro doit être associé le bouton. Sauvez le fichier toujours sous le même nom **VB1G*g*B*bb***. Essayez de trouver le bon nombre de chaque véhicule qui est à commander.

1	**Taux d'intérêt annuel**	10,00%			
2	**Durée du prêt (en mois)**	48			
3					
4		**Camionnette**	**Limousine**	**Compacte**	
5	**Prix**	24.000 €	21.000 €	16.000 €	
6	**Nombre de passagers**	9	6	4	Quantité de Limousine
7	**Quantité à acquérir**	2	0	2	
8					
9	**Montant à emprunter**	48.000 €	0 €	32.000 €	
10	**Mensualité**	1.217,40 €	0,00 €	811,60 €	
11	**Total passagers**	18	0	8	
12					
13	**Montant total des emprunts**	80.000 €			
14	**Total des mensualités**	2.029,01 €			
15	**Nombre total de passagers**	26			
16					

CHAPITRE 4 COMPARAISON ENTRE LDA ET EXCEL VBA

4.1 TYPES DE VARIABLES

En LDA :

`Variable` ou `Tableau`

`Numerique` ou `Caractere` ou `Logique`

Exemple de déclaration de variables :

```
Variable S, i : numérique
Variable Nom, prenom : caractère
Variable Trouve : logique
Tableau  Pays(200) : caractère
Tableau  Nhabitants(200) : numérique
```

Macros VBA:

Les données ne doivent pas être obligatoirement déclarées. Dans un programme complexe, il est préférable de le faire. Nous le ferons toujours ici sauf dans les extraits de programmes. Le nom des variables peut aller jusqu'à 255 caractères. Il ne doit pas commencer par un chiffre et ne doit pas comporter les caractères suivants: espace, "@", "$", "#", ".", "!". Commencer le nom par une lettre (de préférence majuscule).
Lors d'une procédure, les variables servent à stocker toutes sortes de données (des valeurs numériques, du texte, des valeurs logiques, des dates ...). Elles peuvent également faire référence à un objet. Suivant les données que la variable recevra, on lui affectera un type différent. Les différents types de variables de VB sont :

Type de données:	Mot clé :	Espace occupé	Plage de valeur
Octet	Byte	1 octet	Entier de 0 à 255
Logique	Boolean	2 octets	True ou False
Entier	Integer	2 octets	Entier de -32 768 à 32 768
Entier Long	Long	4 octets	Entier de -2 147 483 648 et 2 147 483 647 à 2 147 483 648 et 2 147 483 647
Décimal simple	Single	4 octets	-3,402823E38 à -1,401298E-45 pour les valeurs négatives, 1,401298E-45 à 3,402823E38 pour les valeurs positives.
Décimal Double	Double	8 octets	-1,79769313486E308 à -4,940656458412E-324 pour les valeurs négatives 4,94065645841247E-324 et 1,79769313486231E308 pour les valeurs positives
Monétaire	Currency	8 octets	de -922 337 203 685 477,5808 et 922 337 203 685 477,5807
Date	Date	8 octets	1er Janvier 100 au 31 décembre 9999
Decimal	Decimal	12 octets	+/-79 228 162 514 264 337 593 543 950 335 sans point décimal

			+/-7,9228162514264337593543950335 avec 28 décimales.
Objet	Object	4 octets	toute référence à des objets
Chaîne de caractères à longueur variable	String	10 octets + longueur de chaîne	de 0 à 2 milliards de caractères
Chaîne de caractères à longueur fixe	String	Longueur de la chaîne	1 à 65 400 caractères
Variant avec chiffres	Variant	16 octets	Valeur numérique jusqu'au type double.
Variant avec caractères	Variant	22 octets + longueur de la chaîne	Même plage que pour un String de longueur variable
Défini par l'utilisateur	Type	Variable	Identique au type de données.

Pour rendre obligatoire la déclaration de variables, placez l'instruction "Option Explicit" sur la première ligne du module ou cochez l'option "Déclaration des variables obligatoires" dans le menu "Outils-Options" de l'éditeur de macros.
La déclaration explicite d'une variable se fait par le mot Dim (abréviation de Dimension). La syntaxe est "Dim NomDeLaVariable as Type".

Déclaration explicite: (recommandé)

```
Dim n As Long
Dim chaine As String
Dim logique as Boolean
```

Vous pouvez également déclarer vos variables sur une même ligne :

```
Dim n As Long, Dim chaine As String, Dim logique as Boolean
```

Déclaration par affectation: par défaut de type Variant. (à éviter)

```
n = 0
chaine = "texte initial"
logique = true
```

Déclaration lors de la saisie: (à éviter)

```
n = InputBox( "entrez le nombre n" )
chaine = InputBox( "entrez la chaine de caractère n", , "texte initial")
logique = InputBox("Le tp est-il fini ? (true,false)", "", "true")
```

4.2 EXPRESSIONS ET INSTRUCTIONS D'AFFECTATION

Description	LDA	VBA
Affectation	← (ex: A ← B)	= (ex: A=B)
opérateurs arithmétiques	+, -, *, /	+, -, *, /
	↑ (exponentiation)	^
opérateurs logiques	et	And
	ou	Or
	non	Not
opérateurs relationnels	> , <, ≥, ≤, ≠, =	>, <, >= , <= , <>, =
Opérateur de concaténation	+ (ex: "Mr " + nom + " " + i)	& (ex: "Mr " & nom & " " & i)

4.3 QUELQUES FONCTIONS MATHEMATIQUES

	LDA	VBA
valeur absolue	|x|	=Abs(x)
racine carrée	$\sqrt{x}$	=Sqr(x)
partie entière	⌊ x ⌋	=Int(x)
log népérien, exponentielle	LN(x), EXP(x)	=Log(x), =Exp(x)
Signe		=Sgn(x)
trigonométriques	SIN(x), COS(x), TAN(x), ATAN(x)	=Sin(x), =Cos(x), =Tan(x), =Atn(x)

4.4 INSTRUCTIONS DE LECTURE/ECRITURE

La lecture et l'écriture est assez différente en LDA et dans les macros. En LDA et pour les langages de programmation classiques (FORTRAN, BASIC, PASCAL, C, C++ ...), les instructions sont généralement les mêmes que l'on travaille avec le clavier ou un fichier (en entrée) ou avec un fichier, l'écran ou l'imprimante (en sortie). Avec Excel, les choses sont différentes. En effet, une macro peut communiquer des informations avec les autres feuilles. Ainsi, on peut tout aussi bien entrer des données dans une cellule que dans une boîte de dialogue. Nous y reviendrons plus tard.

	LDA	VBA
Lire (à partir d'un fichier)	`Lire Fichier,...`	Instruction INPUT (voir aide en ligne)
Saisie	`Saisir ...`	Instruction INPUT (voir aide en ligne)
Ecriture (sur fichier)	`Ecrire Fichier,...`	Instruction PRINT (voir aide en ligne)
Impression (imprimante LPT1)	`Imprimer`	Non disponible
Affichage (écran)	`Afficher`	MsgBox(...)

Exemples.

```
Dim Nom As String
Nom = InputBox(" Entrez votre Nom ---> ")
MsgBox " Bonjour " & Nom
```

4.5 STRUCTURE D'ALTERNATIVE

LDA	VBA

Si *condition*	If *condition* Then
alors ...	...
...	...
sinon ...	Else
...	...
Finsi	End If

S'il n'y a pas de clause Else et si le bloc comporte une seule instruction, on peut écrire :

```
If condition Then instruction
```

LDA	VBA
Selon que	If *condition1* Then
condition1 : ...	...
condition2 : ...	ElseIf *condition2*
...	...
conditionK : ...	ElseIf *conditionK*
	...
autrement ...	Else
	...
finselonque	End If

Exemple en **LDA**

```
Variable age : numérique
Afficher "Entrer votre âge"
Saisir age
Si age < 25
alors  Afficher "Vous avez droit à une réduction pour jeunes"
sinon Si age >= 65
      alors Afficher "Vous avez droit à une réduction troisième âge"
      sinon Afficher "Vous payez le prix plein"
      finsi
finsi
Afficher  "Votre âge est de ", age, " ans "
```

VBA

```
Dim Age As Single
Age = InputBox("Entrer votre âge")
If Age < 25 Then
      MsgBox "Vous avez droit à une réduction pour jeunes"
ElseIf Age >= 65 Then
      MsgBox "Vous avez droit à une réduction troisième âge"
Else
      MsgBox "Vous payez le prix plein"
End If
```

4.6 STRUCTURE DE REPETITIVE AVEC CONDITION

Il existe plusieurs structures répétitives en VBA. Nous considérons ici la forme la plus générale qui est aussi la plus pratique dans les langages de programmation.

LDA	VBA
Itérer	Do
bloc d'instructions	*bloc d'instructions*
sortir si condition	If *condition* Then Exit Do
bloc d'instructions	*bloc d'instructions*
Finitérer	Loop

Exemple LDA

```
Variable n,somme,sentinelle,donnee : numérique
n ← 0
somme ← 0
Afficher "Entrez la valeur sentinelle"
Saisir sentinelle
Afficher "Entrez une donnée ou ", sentinelle, " pour terminer"
Itérer
   Saisir donnee
Sortir si sentinelle = donnee
   n ← n + 1
   somme ← somme + donnee
   Afficher "Entrez une donnée ou ", sentinelle, " pour terminer"
finitérer
Afficher "Somme des données= ",somme
Afficher "Nombre de données encodées= ", n
```

Exemple VBA

```
Dim n As Integer
Dim somme As Single, sentinelle As Single, donnee As Single
n = 0
somme = 0
sentinelle = InputBox("Valeur sentinelle")
Do
  donnee = InputBox("Entrez une donnée ou " & sentinelle & _
                    " pour terminer")
If sentinelle = donnee Then Exit Do
   n = n + 1
   somme = somme + donnee
End Do
MsgBox "Somme des données= " & somme
MsgBox "Nombre de données encodées= " & n
```

4.7 BOUCLE AVEC COMPTEUR

LDA	VBA
Pour *compteur* =1 à n faire ... ... finpour	For *compteur* = 1 To *n* ... ... Next *compteur*

On peut aussi effectuer une boucle avec un pas. Dans le cas d'une boucle de pas 1, on omet naturellement la clause "Step".
S'il n'y a qu'une boucle, on peut omettre le nom de la variable de comptage (ici *compteur*) dans l'instruction Next..

LDA	VBA
Pour *compteur* =1 à n par pas de 2 faire ... ... Finpour	For *compteur* = 1 To *n* Step 2 ... ... Next *compteur*

Exemple LDA

```
Variable n, cpteur,fact : numérique
Saisie n
Afficher "Calcul de la factorielle de N. N = ?", n
Fact ← 1
Pour cpteur = 1 à n faire
   Fact ← fact*cpteur
finpour
Afficher n, "!= ", fact
```

VBA

```
Dim n, cpteur As Integer
Dim fact As Double 'C'est bien nécessaire pour un nombre aussi grand
n = InputBox("Calcul de la factorielle de N. N = ?")
fact = 1
For cpteur = 1 To n
   fact = fact * cpteur
Next
MsgBox n & "!= " & fact
```

Remarque. Pour les tableaux, l'indice de début par défaut est 0 au lieu de 1. Pour que l'indice de début par défaut soit 1, il faut mentionner, en dehors d'une procédure :

```
Option Base 1
```

CHAPITRE 5 Exercices de programmation

5.1 Familiarisation avec les programmes en VBA pour EXCEL

Dans le répertoire **MELARD** de l'unité **F:** vous trouverez le classeur Excel « Algos du Cours.xls ».

Faites en une copie dans votre unité **H:** .

Ouvrez ce classeur avec EXCEL.

Il contient neuf feuilles de travail EXCEL. Leurs noms sont ceux des différents sous-chapitres du chapitre 2, partie « Algorithmique » du cours, à l'exception de la feuille « params » qui contient les données relatives à l'un des programmes VBA. Sauvez le classeur sous le nom **VB2G*g*B*bb*.xls**.

Ouvrez l'éditeur « Visual Basic ». Vous le trouverez dans le sous-menu « Macro » du menu « Tools » .

Les modules

Dans la partie gauche de la fenêtre vous pouvez visualisez l'arborescence des différentes catégories « d'objets » qui se trouvent dans le classeur. Il y a essentiellement les feuilles de travail, des formulaires (forms) et des modules. Nous reviendrons plus loin sur les formulaires.

Les modules contiennent le code des programmes VBA. Pour faciliter l'usage de ce classeur, nous avons créé huit modules, chacun de ces modules correspond à une feuille de travail.

Les programmes qui peuvent êtres exécutés à partir d'une feuille de travail sont donc tous écrits dans le module correspondant à la feuille. Ceci n'est pas une obligation : toutes les fonctions et procédures écrites dans n'importe quel module d'un classeur sont accessibles à partir de toute feuille de travail de ce classeur. Dans le classeur « Algos du Cours » on aurait donc peu écrire toutes les procédures et fonction VBA dans un seul module. La répartition en plusieurs modules est laissée à l'appréciation du développeur. Le bon sens de celui-ci et son souci de faciliter l'utilisation le guident pour organiser les modules.

Parcourez successivement toutes les feuilles de travail et exécuter les procédures en cliquant sur les boutons qui s'y trouvent. Lisez dans les modules correspondants le code des procédures exécutées. Efforcez-vous de comprendre l'effet de chaque instruction VBA exécutée. Ceci peut vous aider tant pour comprendre les algorithmes que pour vous familiariser à la programmation VBA. Sauvez souvent le classeur sous le nom **VB2G*g*B*bb*.xls** afin d'éviter de perdre le travail réalisé. Voici la liste des 8 modules, le nombre d'algorithmes, les pages du cours et ce qu'il faut faire :

Séquence	2	page 39	changer les données
Alternative	4	pages 40-42	changer les données
Répétitive	3	pages 44-45	changer un peu les programmes
Compteur	4	pages 46-47	changer les textes affichés
Accumulateur	2	pages 48-49	changer les noms de quelques variables
Complet	1	page 53	corriger le cas où on entre 0 la première fois
Tableau	6	pages 55-58	1. changer les dernières données et le nombre 2. simplement exécuter les autres : changer les noms des pays
Module	1	page 59	changer les données des deux matrices

Si vous voulez tester les exemples du cours plus à votre aise, n'oubliez pas de prendre une copie sur disquette du classeur.

Les formulaires utilisateur (userforms)

Dans les chapitres précédents, vous avez appris à utiliser les boîtes de message (MsgBox) et les boîtes de saisie (InputBox).

Lorsque l'on doit saisir plusieurs données, éventuellement de types différents, utiliser une succession de boîtes de saisie peut être inconfortable et obscurcir l'utilisation d'un développement. Il vaut mieux alors utiliser un formulaire utilisateur. Celui-ci permet de saisir dans une seule fenêtre toutes les données. Dans le formulaire, le développeur peut également afficher toute l'information utile à l'utilisateur. Voir la feuille « Alternatives », cliquer sur le bouton « Formulaire ».

Nous ne vous demanderons pas de créer des formulaires utilisateur dans les exercices qui suivront, mais il est bon de savoir qu'ils existent.

5.2 Programme à corriger et à exécuter

Il s'agit d'un programme effectuant un procédé élémentaire de cryptographie. La cryptographie permet de transmettre des messages qui peuvent être interceptés sans pouvoir révéler leur contenu. De tels algorithmes sont de nos jours employés sur l'Internet pour transmettre des informations bancaires et financières, par exemple.

Pour simplifier le document à transmettre est composé seulement de chiffres, par exemple:

`12345678901234567890123456789O` (30 chiffres)

L'expéditeur spécifie une clé de 8 chiffres connue du destinataire, par exemple: `29022002` (pour le retenir: 29 février 2002)

Le cryptage consiste à juxtaposer la clé autant de fois que nécessaire ici quatre fois (4x8 = 32 > 30):

`29022002290220022902200229022002`

et à ajouter chiffre à chiffre modulo 10, pour obtenir le message

clé : `29022002290220022902200229022002`

document : `12345678901234567890123456789O`

message : `31367670191454589792323675701O`

par exemple: chiffre 1: 2 + 1 = 3
chiffre 2: 9 + 2 = 11 donc 1
chiffre 3: 0 + 3 = 3
chiffre 29: 2 + 9 = 1
chiffre 30: 0 + 0 = 0

Le destinataire reçoit le message et doit recomposer le document.

Il procède par soustraction message - clé, chiffre par chiffre, modulo 10, donc en ajoutant 10 si la différence est strictement négative :

message : `31367670191454589792323675701O`

clé : `29022002290220022902200229022002`

résultat : `12345678901234567890123456789O`

par exemple: chiffre 1: 3 - 2 = 1

chiffre 2: 1 - 9 = -8 donc 2
chiffre 3: 3 - 0 = 3
chiffre 29: 1 - 2 = -1 donc 9
chiffre 30: 0 - 0 = 0

Dans votre unité **H:**, faites une copie du classeur « CryptoVBA.xls » qui se trouve dans le répertoire **MELARD** de l'unité **F:**. Ouvrez-le et sauvez-le chez vous sous le nom **VB3G*g*B*bb*.xls**.

Le classeur contient une feuille de travail et un module VBA :

- feuille Crypto, module ModuleCrypto.

Essayez le programme avec une longueur de 9. Vous devrez corriger le programme VBA qui est la traduction en VBA de l'algorithme contenu sous forme de commentaires dans le module. La feuille de travail et le module VBA contiennent toutes les informations nécessaires à l'exécution du travail. Efforcez-vous de bien comprendre chaque ligne du code VBA.

Indications. L'algorithme est correct mais il y a 2 erreurs dans le programme. Notez que pour simplifier l'utilisation, on ne saisit pas la clé mais on la définit dans le programme. Pour éviter de saisir le message plusieurs fois, vous pouvez procéder d'une manière similaire. Nous avons déjà simplifié la sortie pour permettre une vérification plus aisée.

5.3 Ecriture de programmes en VBA

5.3.1 Banques

Dans cette section, nous allons voir comment créer un programme en VBA (un exemple de la programmation modulaire). On va utiliser un mélange de fonctions et de procédures. L'exemple qu'on va considéré est celui que vous avez déjà vu au chapitre 3 des exercices sur Excel. Il s'agit du problème des banques d'information.

On dispose des données relatives aux coûts de connexion à certaines banques d'informations ainsi que des données indiquant pour des utilisateurs de ces banques les durées de connexion et d'autres paramètres entrant dans le calcul du coût d'une connexion (nombre de références consultées, plage horaire où se situe la connexion, etc.).

La banque A:

Le coût d'une connexion est proportionnel à sa durée (exprimée en minutes) et tient compte du nombre de références consultées selon la formule suivante:

coût = durée * prix_minute + nb_ref * prix_ref

La banque B:

On y définit trois prix différents en fonction de la plage horaire où se situe la connexion. De ce fait, le prix de la connexion est obtenu par la formule suivante:

```
coût =   durée_1 * prix_1
       + durée_2 * prix_2
       + durée_3 * prix_3
```

La banque C:

Cette banque d'information pratique un taux régressif en fonction de la durée de la connexion: 100% du prix de base pour les dix premières minutes, 90% pour les dix minutes suivantes, 75% pour les dix minutes suivantes et seulement 55% pour le reste du temps de connexion.

Rappel de l'algorithme utilisé dans Excel: *d* désigne la durée de connexion et *p* le prix de base d'une minute.

```
Si d < 10
      coût = d * p
sinonsi d < 20
      coût =  10 * p + (d-10) * p * 0.9
sinonsi d < 30
      coût =  10 * p + 10 * p * 0.9 + (d-20) * p * 0.75
sinon
      coût =  10 * p + 10 * p * 0.9 + 10 * p * 0.75 + (d-30) * p * 0.55
finsi
```

La banque D:

Pour y avoir accès, il faut y être abonné et avoir payé une redevance (différente selon les utilisateurs). Cette redevance représente un avoir mensuel (non cumulable) en temps de connexion à cette banque d'informations car seules les minutes de connexions supplémentaires sont facturées au tarif indiqué. Ainsi, un utilisateur qui s'est connecté pendant une durée donnée exprimée en minutes n'aura rien à payer si la relation

durée * prix_minute ≤ redevance

est vérifiée et ne payera que la différence dans le cas contraire.

Notez qu'il existe dans VBA une fonction "max", comme dans Excel, qui vous permettra de déterminer élégamment et simplement la somme à payer (allez voir dans votre fichier Excel la formule que vous avez utilisé pour déterminer le coût de la Banque D)

On se proposait dans l'exercice Excel de réaliser un certain nombre de tâches dont nous ne retiendrons que les suivantes (*ne commencez pas maintenant; lisez plutôt l'énoncé jusqu'au bout*):

(1) Calculer, pour chaque utilisateur, le montant dû pour les connexions qu'il a effectuées à chacune des banques d'informations durant une période donnée (un mois pour fixer les idées).

(2) Calculer le montant global dû pour chaque utilisateur

(3) Calculer la somme due à chaque banque par l'ensemble des utilisateurs, présenter ces sommes dans un graphique de type histogramme

La feuille de calcul était composée de deux tableaux. Le premier contenait les coûts de connexions par utilisateur et par banque d'informations ainsi que les totaux associés (nous considérions un exemple avec 7 utilisateurs) et se présentait comme suit:

TABLEAU 1. COUTS DES CONNEXIONS AUX BANQUES D'INFORMATIONS

	Banque A	Banque B	Banque C	Banque D	Total
Ut 1	xxxx.xx	xxx.xx	xxxx.xx	xxxx.xx	xxxxx.xx
Ut 2	xxx.xx	xxx.xx	xxxx.xx	xxx.xx	xxxxx.xx
...					
Ut n					
TOTAL	xxxxx.xx	xxxxx.xx	xxxxx.xx	xxxx.xx	xxxxx.xx

Le deuxième tableau contenait les données du problème (tarifs des connexions et informations relatives aux utilisateurs) ainsi que la moyenne des utilisations par banque lors de la période précédente. Ce tableau ressemblait à ceci :

TABLEAU 2. TARIF DES CONNEXIONS AUX BANQUES D'INFORMATIONS

Banques	A		B			C	D
prix	min	ref	min_1	min_2	min_3	min	min
	30.5	46.8	22.6	35.7	52.4	48.2	32.6

RECAPITULATIF DES DUREES DE CONNEXION

Banques	A		B			C	D	
Usagers	durée	nb ref	durée	durée	durée	durée	d	red
Ut 1	152	327	80	54	40	143	120	3450
Ut 2	328	290	120	34	38	0	0	0
...								
Ut n								

M-PRECED
M-ACTUEL

Au lieu d'employer Excel, on envisage d'écrire un programme modulaire en VBA.

Une partie du programme est écrite dans le module "Modulebanques". Le traitement est découpé en procédures et fonctions dont les noms sont les suivants :

Sub Banques ( )
Procédure principale

Function SaisieTarifs ( )
On saisit les données du tarif pour les quatre banques

Function NomUtilisateur ( )
On saisi le nom d'un utilisateur

Function AfficheDepUtilisateur (Utilisateur, DepMois)
On affiche le nom de l'utilisateur et le total des dépenses du mois actuel d'un utilisateur donné.

Function CoutBanqueA (PminA, PrefA)
Il s'agit de saisir la durée de connexion, duree, et le nombre de références, nbref, d'un utilisateur de la banque A et de calculer le coût correspondant:

CoutBanqueA = PminA * duree + PrefA * nbref

Function CoutBanqueB (P1B,P2B,P3B)
Il s'agit de saisir les trois durées de connexion duree1, duree2 et duree3 pendant les trois types de plages horaires et de calculer le coût total correspondant

Function CoutBanqueC (PC)
Ici, on saisit la durée de connexion d'un utilisateur à la banque C, duree, et on calcule le coût en se basant sur le tarif dégressif suivant: 100 % pour les 10 premières minutes, 90 % pour les 10 minutes suivantes, 75 % pour les 10 minutes suivantes et 55 % pour le reste du temps de connexion

Function CoutBanqueD (PminD)
Toujours pour un utilisateur, on saisit la durée de connexion à la banque D, duree, et le montant de redevance qu'il a payée, redevance, qui représente un avoir mensuel (non cumulable) en temps de connexion car seules les minutes de connexion supplémentaires sont facturées au tarif indiqué; l'utilisateur n'aura rien à payer si

duree*PminD < redevance

A titre d'exemple, la fonction CoutBanqueA se présentera comme suit:

```
Function  CoutBanqueA (PminA as Double, PrefA as Double) as Double
  Dim  Duree as Double, NbRef as Double
  Duree = 0
  Nbref = 0
  Duree = InputBox ("Banque A, durée de connexion : ")
  NbRef = InputBox  ("Banque A, nombre de références : ")
  CoutBanqueA = Duree*PminA + NbRef*PrefA
End Function
```

Les variables créées dans la fonction CoutBanqueA sont locales à la fonction, comme Duree et NbRef.

Travail à faire :

On demande

1. Dans votre unité **H:** , faites une copie du classeur « BanquesVBA.xls » qui se trouve dans le répertoire **MELARD** de l'unité **F:**. Ouvrez-le et sauvez-le chez vous sous le nom **VB4G*g*B*bb*.xls**. Le classeur contient une feuille de travail (Banques) et un module VBA (ModuleBanques).
2. Dans le module Banques, vous devrez écrire un programme VBA (ensemble de procédures et fonctions) qui est la traduction en VBA des algorithmes contenus sous forme de commentaires dans le module.
3. D'écrire dans le module « ModuleBanques » la fonction BanqueA telle que présentée si dessus en y ajoutant les instructions nécessaires pour s'assurer que Duree et NbRef soient supérieurs ou égaux à 0 ; de tester le programme;
4. Toujours dans le même module, de détailler les fonctions relatives aux trois autres banques;
5. De tester votre programme avec les données utilisées pour l'exercice Excel.

5.3.2 La billetterie du parc d'attraction WILABA

Votre tâche est d'écrire un programme pour traiter la billetterie journalière dans le parc d'attraction 'WILABA' ouvert de 10h à 20h.

Il n'y a pas de droit d'entrée, mais 10 attractions sont payantes.

Toutes ces attractions durent 20 minutes. Une première séance commence à 10h et se termine à 10h20, après une pose de 10 minutes commence la séance suivante et ainsi de suite jusqu'à 20h. Les heures de séances sont donc 10h, 10h30, 11h, 11h30, 12h, 12h30, 13h, 13h30, 14h, 14h30, 15h, 15h30, 16h, 16h30, 17h, 17h30, 18h, 18h30, 19h, 19h30, Au cours d'une journée il y a donc 20 séances durant lesquelles fonctionnent chacune des 10 attractions payantes.

Chaque attraction a sa capacité propre de spectateurs. La première (la plus grande) 'Teteenbas' contient *GBB* places (c'est-à-dire *150* places pour le *binôme 50* du *groupe 1*). Le nombre de places exact (minimum 60) des autres, ainsi que leur noms, est laissé à votre appréciation.

Le tarif est le même pour toutes les attractions (10 EUR sans réduction et 6 EUR avec la carte avantage Wilaba+bonus+). On peut prendre des billets pour une attraction durant les 30 minutes qui précèdent la séance. En fait l'algorithme ne contiendra pas la notion de temps, mais devra permettre à l'aide d'une valeur sentinelle, introduite par la caissière, de passer d'une séance à l'autre (par

exemple: en consultant sa montre à 11h30, la caissière introduit cette valeur sentinelle pour clôturer la vente de billets pour la séance 4 et commencer la vente pour la séance 5.

Le programme doit, pour chaque demande de billet(s), saisir le numéro d'attraction, le nombre de places souhaité et le tarif appliqué (on suppose un seul tarif appliqué par demande). Il affichera ensuite le prix dû par le client. Le programme doit également mettre à jour le nombre de places libres par attraction et prévenir le client lorsqu'il n'y a plus de place ou presque (moins de 20 places) afin d'éviter de placer des gens aux endroits qu'ils ne souhaitent pas. La recette totale de la journée doit également être mise à jour.

A l'heure de changer de séance, la caissière entre la sentinelle (par exemple une valeur négative de numéro d'attraction). Le programme doit alors afficher à l'écran la recette par attraction. La recette totale et la fréquentation des attractions (en pourcentage) seront affichées lorsque les 20 séances auront été clôturées.

Vous rédigerez l'algorithme sous forme de commentaire dans un module VBA dans un fichier sauvé sous le nom **VB5G*g*B*bb*.xls**, où ***g*** est votre numéro de groupe et ***bb*** est votre numéro de binôme. Vous le traduirez en VBA. Nous vous demandons bien sûr de le tester.

5.3.3 M. Olaire, dentiste (Examen d'algorithme de juin 2000)

M. Olaire est dentiste. Il a informatisé une partie de la gestion de son cabinet dans Excel mais veut aller plus loin. Ses premiers essais sont infructueux au point qu'il s'arrache les cheveux (pas ses dents heureusement). Comme son fils a appris l'informatique à l'Université, il lui demande de l'aider. Celui-ci n'est pas en sciences économiques et n'a pas de connaissances en algorithmique. Il vous demande donc de réaliser l'algorithme pour lui, en langage de description d'algorithme (vous pouvez écrire le programme directement en VBA si vous préférez, mais veuillez ne pas mélanger). Les patients sont identifiés par leur numéro de sécurité sociale (de 8 chiffres exactement). Un fichier PATIENTS.TXT sera lu en mémoire par la procédure `LecturePatients` que vous ne devez pas écrire (vous pouvez donc supposer que les données sont déjà en mémoire). Il comporte, pour chaque patient, les informations numériques suivantes: son numéro de sécurité sociale et la date de sa dernière visite (sous forme de nombre de jours écoulés depuis le 1er janvier 1980). Le numéro de ligne est le numéro d'inscription chez M. Olaire (entre 1 et 2000). Dans un autre fichier ADRESSES.TXT qui sera également en mémoire grâce à la procédure `LectureNomPrAdresses`, la ligne *i* contient le nom, le prénom et l'adresse du *i*ème patient de M. Olaire (sur 128 caractères).

L'algorithme lit les deux fichiers (voir ci-dessous) et crée trois tableaux à une dimension appelés assez naturellement: NumSS, DateDernVisite et NomPrAdresse. Ils sont de type numérique sauf ce dernier. Notez que le numéro de patient n'est pas enregistré dans un tableau puisqu'il n'est pas nécessaire. A partir d'une certaine ligne, les éléments sont tous nuls parce que M. Olaire n'a pas encore 2000 patients. L'algorithme commence par afficher de manière répétée un menu offrant de quitter le programme ou d'effectuer une des deux actions suivantes.

1. Mettre à jour la date de dernière visite. Le dentiste doit saisir le numéro de sécurité sociale. L'algorithme refuse absolument un numéro de sécurité sociale qui ne comporte pas exactement 8 chiffres. Quand le numéro est accepté, il consulte le tableau NumSS et détermine la ligne *i* sur laquelle se trouvent les informations relatives au patient. L'élément correspondant de DateDernVisite est alors remplacé par la date du jour (voir ci-dessous). Si le numéro de sécurité sociale n'existe pas dans le tableau NumSS, c'est que le patient n'est pas encore inscrit chez M. Olaire et cette inscription est alors effectuée en remplaçant la première ligne contenant des zéros des tableaux NumSS, et DateDernVisite et en complétant la même ligne du tableau NomPrAdresse. Il faut évidemment signaler si jamais il n'y avait plus de place dans le tableau.

2. Produire une lettre de rappel pour tous les patients dont la dernière visite date de plus de 180 jours. Ce nombre de jours s'obtient en calculant la différence entre l'élément de DateDernVisite et la date du jour (voir ci-dessous). Cette lettre se présente comme ceci (n'écrivez les instructions Imprimer que pour les lignes intéressantes, celles indiquées entre parenthèses contenant de l'information variable selon le patient)

   "(nom, prénom et adresse)
   Chère Madame, Mademoiselle, Monsieur,
   Puis-je vous faire observer que votre dernière visite remonte à il y a (nombre de jours) jours.
   Formule de politesse
   M. Olaire"

On ne demande pas d'écrire la version mise à jour des deux fichiers PATIENTS.TXT et ADRESSES.TXT. Trois procédures sont fournies. Pour les utiliser, employez les instructions suivantes aux endroits appropriés de votre algorithme:

```
Effectuer LecturePatients( NumSS, DateDernVisite )
Effectuer LectureNomPrAdresses( NomPrAdresse )
Effectuer LectureDateDuJour( NJoursEcoulés )
```

Indication. On conseille de déterminer le nombre de patients NbPat au début de l'algorithme.

On demande

1. Dans votre unité **H:** , faites une copie du classeur « MOlaireVBA.xls » qui se trouve dans le répertoire **MELARD** de l'unité **F:**. Ouvrez-le et sauvez-le chez vous sous le nom **VB6G*g*B*bb*.xls**. Le classeur contient des feuille de travail correspondant aux deux fichiers mentionnés (PATIENTS.TXT et ADRESSES.TXT).

2. Dans un module, vous devrez écrire l'algorithme sous forme de commentaires. Un corrigé de l'algorithme sera disponible sur l'université virtuelle juste avant la dernière séance de VBA.

3. Vous devrez ensuite écrire un programme VBA (ensemble de procédures et fonctions) qui est la traduction en VBA de l'algorithme contenu sous forme de commentaires dans le module.

4. De tester votre programme avec les données fournies.

Résolution de l'exercice 5.3.3 M. Olaire, dentiste

N.B. Nous donnons ici une solution modulaire. Une solution non modulaire était aussi acceptée.

```
Algorithme "Gestion du cabinet de M. Olaire, dentiste"
    Tableau NumSS(2000), DateDernVisite(2000) : numérique
    Tableau NomPrAdresse(2000) : caractère
    Variable NbPat : numérique
    Effectuer LecturePatients (NumSS, DateDernVisite)
    Effectuer LectureNomPrAdresses (NomPrAdresse)
    Effectuer CalculNombrePatients(NumSS, NbPat)
    Effectuer AffichageMenu(NumSS, DateDernVisite, NomPrAdresse, NbPat)
finalgo

Procédure CalculNombrePatients(NumSS : entrée, NbPat : sortie)
    Tableau NumSS(2000) : numérique
    Variable j, NbPat : numérique
    j <- 0
    NbPat <- 2000
    Itérer
        j <- j + 1
    sortirsi NumSS(j) = 0 ou j = NbPat
    finitérer
    Si NumSS(j) > 0 alors
        Afficher 'Plus de place dans le tableau'
    sinon
        NbPat <- j - 1
    finsi
finprocédure

Procédure AffichageMenu(NumSS, DateDernVisite, NomPrAdresse, NbPat : entrée-sortie)
    Variable NbPat : numérique
    Tableau Numss(2000), DateDernVisite(2000) : numérique
    Tableau NomPrAdresse(2000) : caractère
    Itérer
        Afficher "Tapez 1 pour mettre à jour la dernière date de visite"
                Afficher "Tapez 2 pour envoyer une lettre de rappel "
                Afficher "Tapez 0 pour sortir "
        Saisir n
    sortirsi n = 0
        Si n = 1 alors
                        Effectuer TraitementMiseAJour(NumSS, DateDernVisite, NomPrAdresse,
NbPat)
                sinon
                        Effectuer ProductionLettreDeRappel(NumSS, DateDernVisite,
NomPrAdresse, NbPat)
        finsi
    finitérer
finprocédure

Procédure TraitementMiseAJour(NumSS, DateDernVisite, NomPrAdresse, NbPat : entrée-sortie)
    Variable numéro, i, NJoursEcoulés, NbPat : numérique
```

```
    Variable trouvé : logique
    Tableau NumSS(2000), DateDernVisite(2000) : numérique
    Tableau NomPrAdresse(2000) : caractère
    Effectuer SaisieNumSS(numéro)
    Effectuer LectureDateDuJour(NJoursEcoulés)
    i <- 0
    trouvé <- faux
    Itérer
        i <- i + 1
                trouvé <- numéro = NumSS(i)
    sortirsi trouvé ou i = NbPat
    finitérer
    Si trouvé alors
        DateDernVisite(i) <- NJoursEcoulés
    sinon
        Si NbPat < 2000 alors
                NbPat <- NbPat + 1
                        NumSS(NbPat) <- numéro
                DateDernVisite(NbPat) <- NJoursEcoulés
            Afficher "Entrez le nom, prénom et adresse du patient"
                Saisir NomPrAdresse(NbPat)
        sinon
                Afficher "Plus de place dans le tableau. Le patient n'est pas ajouté"
        finsi
    finsi
finprocédure

Procédure SaisieNumSS(numéro : sortie)
    vvVariable numéro : numérique
        Afficher "Entrez le numéro de sécurité sociale du patient"
    Itérer
        Saisir numéro
    sortirsi numéro > 1 et numéro < 100000000 et numéro = |_numéro_|
        Afficher "Erreur. Le numéro de sécurité sociale doit être entre 1 et 99999999"
    finitérer
finprocédure

Procédure ProductionLettreDeRappel(NumSS, DateDernVisite, NomPrAdresse, NbPat : entrée)
    Variable nombre_de_jours, i, NJoursEcoulés, NbPat : numérique
    Tableau NumSS000), DateDernVisite(2000) : numérique
    Tableau NomPrAdresse(2000) : caractère
    Effectuer LectureDateDuJour(NJoursEcoulés)
    Pour i = 1 à NbPat
        nombre_de_jours <- NJoursEcoulés - DateDernVisite(i)
                Si nombre_de_jours > 180 alors
                Imprimer NomPrAdresse(i)
                Imprimer "Votre dernière visite remonte à il y a ", nombre_de_jours, " jours"
        finsi
    finpour
finprocédure
```