

## Séance 3

# PROGRAMMATION VBA: Fonctions et procédures, variables, tests et boucles

L3

Management / Comptabilité Contrôle Finances/ Économie de la Firme

Université de Bordeaux

- I. Le langage VBA
- II. Les variables, les procédures, les fonctions
- III. Les tests et les boucles
- IV. Entrées, sorties, objet feuille

## 1. Le Visual Basic et le VBA

- Le Visual Basic est un dérivé du langage Basic, qui permet d'avoir un aperçu de l'interface avant de l'exécuter.
- Le VBA (Visual Basic for Applications) constitue une implémentation du VB au sein des applications de la suite MicrosoftOffice.

## 2. Mise en œuvre de programmes VBA dans Excel

- Aller dans l'onglet Fichier, puis Options, Personnaliser le Ruban et cocher la case Développeur.

# I. Le langage VBA

- Il existe deux possibilités pour écrire et enregistrer un programme VBA dans Excel:
  - Onglet Développeur, bouton Macro, puis saisir un nom et cliquer sur Créer: l'interface du développeur VBA s'ouvre avec une procédure du nom donné précédemment;
  - Onglet Développeur, bouton Visual Basic: l'interface du développeur VBA s'ouvre, avec le bloc note vide (ou **Alt+F11**);
  - On peut également faire Enregistrer une macro, puis visualiser le code correspondant dans l'interface du développeur VBA.

## 2. Algorithmes et programmation

- Un algorithme est la description de la suite des opérations élémentaires ordonnées capables de résoudre le problème posé.
- Un programme est la traduction d'un algorithme en un langage compréhensible par l'ordinateur.
- En principe, la démarche à suivre consiste dans un premier temps à concevoir un algorithme, puis à le traduire dans le langage de programmation souhaité
- Un algorithme doit être facilement transposable dans tout type de langage de programmation

## 3. Exemple d'algorithme

On veut construire un programme permettant de calculer automatiquement le prix TTC d'un article, connaissant son prix HT, la TVA étant donnée à 20%.

- On identifie les variables du problème:
  - Le prix HT, que l'on peut noter  $p_{HT}$
  - La valeur de la TVA, que l'on peut noter TVA
  - Le prix TTC, que l'on peut noter  $p_{TTC}$

# I. Le langage VBA

- Le calcul du prix TTC est donné par :  
$$pTTC = (1 + TVA/100) * pHT$$
- On peut construire l'algorithme du calcul du prix TTC:

## Début de l'algorithme

**Variables:** Décimaux: pTTC;pHT;TVA

TVA  $\leftarrow$  20

Lire pHT

pTTC  $\leftarrow (1 + TVA/100) * pHT$

Afficher pTTC

## Fin de l'algorithme

# I. Le langage VBA

- Remarque: avec AlgoBox, cela donne:

1 **VARIABLES**

2   pHT EST\_DU\_TYPE NOMBRE

3   pTTC EST\_DU\_TYPE NOMBRE

4   TVA EST\_DU\_TYPE NOMBRE

5 **DEBUT\_ALGORITHME**

6   TVA PREND\_LA\_VALEUR 0.2

7   LIRE pHT

8   pTTC PREND\_LA\_VALEUR  $(1 + TVA/100) * pHT$

9   AFFICHER pTTC

10 **FIN\_ALGORITHME**



# II. Les variables, les procédures, les fonctions

## 1. Les variables

- En programmation, une variable est un nom qui sert à repérer un emplacement donné de la mémoire centrale.
- Les variables permettent de manipuler les valeurs sans avoir à se préoccuper de l'emplacement qu'elles occupent effectivement en mémoire.
- Choix des noms des variables : doit être le plus parlant possible.
- Exemple : pTTC, TVA, pHT, c'est mieux que d'écrire x,y,z.

## II. Les variables, les procédures, les fonctions

- Une variable = {identificateur, type, valeur}
  - Identificateur : nom par lequel la variable est manipulée dans le programme
  - Type : type des valeurs possibles que la variable peut contenir
  - Valeur : valeur stockée dans la variable

# II. Les variables, les procédures, les fonctions

- Identificateur d'une variable:
  - Elle doit commencer par un caractère alphabétique et ne pas comporter les caractères suivants : . % , + - \* ! # @ \$
  - Elle ne peut pas excéder 255 caractères.
  - Pour faciliter la lisibilité des programmes, on s'efforcera d'utiliser des minuscules pour les variables utilisées par le programme.
- Déclaration des variables:
  - Par défaut, il n'est pas nécessaire de déclarer les variables utilisées. On peut donc utiliser n'importe quelle variable sans se préoccuper de sa déclaration préalable
  - Les variables sont alors de type Variant :  $\Rightarrow$  pas d'optimisation
  - Il vaut mieux déclarer explicitement les variables avec la syntaxe : Dim identificateur As Type

## II. Les variables, les procédures, les fonctions

- Option Explicit : à placer en début de module, avant la première macro pour rendre obligatoire la déclaration de toutes les variables.
- Les types de variables acceptés dans VBA:
  - **Integer** : entier court entre -32768 et +32767;
  - **Long** : entier long entre -2147483648 et +2147483647;
  - **Single** : réel simple précision, 7 chiffres significatifs max;
  - **Double** : réel en double précision, 15 chiffres significatifs max;
  - **Currency** : nombre monétaire ;
  - **Date** : date et heure;
  - **String** : chaîne de caractères ;
  - **Boolean** : booléen (*True* ou *False*) ;
  - **Object** : référence quelconque à un objet ;
  - **Variant** : type particulier pouvant être n'importe quel autre type (comme le format Standard d'Excel).

## II. Les variables, les procédures, les fonctions

- Outre ces types élémentaires, il est également possible de créer des tableaux et des types personnalisés (voir chapitre 4).
- Exemple de déclaration des variables de l'algorithme vu précédemment:

Dim pTTC As Single

Dim pHT As Single

Dim TVA As Single

Ou bien:

Dim pTTC As Single, pHT As Single, TVA As Single

Mais pas: Dim pTTC, pHT, TVA As Single car dans ce cas, pTTC et pHT reçoivent le type Variant !

## II. Les variables, les procédures, les fonctions

- Les **constantes**

Const TVA As Integer=20

Ou bien: Const TVA=20

## 2. Les fonctions

- Elles renvoient une **valeur unique**. Cette valeur est représentée par le nom de la fonction: il faut donc préciser le type de variable.

Syntaxe: Function Nom\_fonction (variable(s) As Type) As Type  
End Function

- Exemple : fonction Prix\_TTC permettant de calculer le prix TTC, connaissant le prix HT:

Const TVA As Integer =20

Function Prix\_TTC (pHT As Single) As Single

Prix\_TTC=pHT\*(1+TVA/100)

End Function

## II. Les variables, les procédures, les fonctions

- Les fonctions ainsi créées en VBA peuvent être appelées dans une feuille de calcul, au même titre que les autres fonctions de l'application Excel.
- Attention: pour sauvegarder une macro (une fonction par exemple), il faut enregistrer la feuille de calcul sous le format .xlsm (Classeur Excel prenant en charge les macros)
- Si un programme bloque, faire **Ctrl + Pause (Break)**



## 3. Les procédures

- Contrairement aux fonctions, les procédures (subroutines) sont des macros qui ne renvoient pas de valeur.
- Syntaxe : Sub Nom\_Procédure()  
End Sub
- Les arguments à l'intérieur de la parenthèse ne sont pas obligatoires, mais même sans arguments, il faut placer les parenthèses !
- Si on veut exécuter un des programmes, on place le curseur à l'intérieur du bloc et on clique sur exécuter. (ou F5)

## II. Les variables, les procédures, les fonctions

- Exemple de procédure sans argument:

Ecrivons une procédure permettant d'afficher un message d'information:

```
Sub Message_Info()  
    Call MsgBox("Vous êtes en cours d'informatique")  
End Sub
```

- Exemple de procédure avec arguments:

Ecrivons une procédure permettant de calculer un taux de réduction faisant suite à deux remises successives.

## II. Les variables, les procédures, les fonctions

```
Sub Taux_Remise (Remise_1 As Single, Remise_2 As Single,  
Remise_finale As Single)  
    Remise_finale = (1 - ((1 - Remise_1 / 100) * (1 - _  
Remise_2 / 100))) * 100  
End Sub
```

Ce type de procédure n'est pas amenée à être directement exécutée, mais à être appelée.

- Remarque: pour changer de ligne au milieu d'une séquence d'instruction, utiliser le signe « \_ »
- En VBA, il faut écrire une instruction par ligne (faire Enter en fin de chaque instruction. Pour écrire deux instructions différentes sur une même ligne, utiliser le signe « : »

## II. Les variables, les procédures, les fonctions

- **Appel de procédures:** l'appel d'une procédure se fait avec *Call*

Exemple d'appel de procédure:

```
Sub Calcul_remise_finale()
```

```
Dim Remise As Single
```

```
    Call Taux_Remise (8, 4, Remise) 'Ici, aucun message ne  
                                     s'affiche
```

```
End Sub
```

Remarque: les commentaires des programmes VBA sont identifiés par le signe '

### 3. Saisie de données et affichage de données en VBA

- La saisie de données se fait par l'instruction *InputBox*
- L'affichage de données se fait par l'instruction *MsgBox*
- Exemple:  
Demander les deux taux de remise successives, puis afficher le taux de remise finale.

## II. Les variables, les procédures, les fonctions

```
Sub Affichage_Remise_Finale()
```

```
Dim Remise_finale As Single, Remise_1 As Single, Remise_2 As Single
```

```
    Remise_1 = InputBox("Entrer le taux de la première  
    remise")
```

```
    Remise_2 = InputBox("Entrer le taux de la deuxième  
    remise")
```

```
    Remise_finale = (1 - ((1 - Remise_1 / 100) * _  
    (1 - Remise_2 / 100))) * 100
```

```
    MsgBox ("La remise finale est égale à " &  
    Remise_finale)  
End Sub
```

## II. Les variables, les procédures, les fonctions

- *Et le même programme avec un appel de procédure:*

```
Sub Affichage_Remise_Finale_2()
```

```
Dim Remise_finale As Single, Remise_1 As Single, Remise_2 As Single
```

```
    Remise_1 = InputBox("Entrer le taux de la première  
    remise")
```

```
    Remise_2 = InputBox("Entrer le taux de la deuxième  
    remise")
```

```
    Call Taux_Remise(Remise_1, Remise_2, Remise_finale)
```

```
    MsgBox ("La remise finale est égale à " &  
    Remise_finale)
```

```
End Sub
```

# III. Les tests et les boucles

## 1. Les tests (structures conditionnelles)

- **If ... Then ...Else ... Elself ... EndIf**

- permet d'écrire une structure conditionnelle

- syntaxe :  
*If <condition> then*  
*<instructions>*  
*Else*  
*<instructions>*  
*End If*

- Dans le cas où il est nécessaire d'imbriquer plusieurs structures conditionnelles, on peut utiliser l'instruction Elself :

- If <condition> Then*  
*<instructions>*  
*Elself <condition>*  
*<instructions>*  
*End If*



# III. Les tests et les boucles

- Exemple: connaître la mention d'un étudiant à un examen

Sub mention()

Dim note As Single

note = InputBox("Quelle est la moyenne générale?")

If note < 10 Then

MsgBox ("ajourné")

Elseif note < 12 Then

MsgBox ("passable")

Elseif note < 14 Then

MsgBox ("assez bien")

Elseif note < 16 Then

MsgBox ("bien")

Else

MsgBox ("très bien")

End If

End Sub

# III. Les tests et les boucles

- **Select...Case**

- Permet d'écrire une structure conditionnelle dans laquelle une expression doit être comparée à plusieurs valeurs.
- La syntaxe est la suivante :

```
Select Case <variable>  
  Case valeur1:  
    <instructions>  
  Case valeur2:  
    <instructions>  
  ...  
  Case Else:  
    <instructions>  
End Select
```

# III. Les tests et les boucles

- Lorsque la variable est égale à une valeur répertoriée, les instructions correspondantes sont exécutées, et l'instruction *Select Case* se termine. La ligne *Case Else* permet d'inclure toutes les occurrences de la variable non répertoriées auparavant. Elle est facultative.

- Exemple 1: réécriture du programme précédent

```
Sub mention_bis()  
Dim note As Single  
note = InputBox("Quelle est la moyenne générale?")  
Select Case note  
    Case Is < 10  
        MsgBox ("ajourné")  
    Case Is < 12  
        MsgBox ("passable")  
    Case Is < 14  
        MsgBox ("assez bien")  
    Case Else  
        MsgBox ("très bien")  
End Select  
End Sub
```

# III. Les tests et les boucles

- Exemple 2:

```
Sub couleur()  
Dim Couleur As String  
Couleur=InputBox("Choisir la couleur rouge, bleu ou vert")  
Select Case Couleur  
    Case "rouge"  
        MsgBox("Vous avez choisi rouge")  
    Case "bleu"  
        MsgBox(" Vous avez choisi bleu")  
    Case "vert"  
        MsgBox(" Vous avez choisi vert")  
    Case Else  
        MsgBox("Choisissez rouge, bleu ou vert")  
End Select  
End Sub
```

# IV. ENTEES ET SORTIES – OBJET FEUILLE

Les fenêtres prédéfinies permettent de saisir ou d'afficher du texte:

**Saisie de texte** : c'est la fonction **InputBox** qui affiche une boîte de saisie et retourne une chaîne de caractères :

Syntaxe : `var1 = InputBox("message pour l'utilisateur ", "Titre de la fenêtre ", " valeur par défaut ")`

Exemple : `dim n as Integer`

```
...  
n = InputBox("Donner un entier ", "Exemple",0)  
...
```

**Affichage de message** : c'est la fonction **MsgBox** qui affiche une boîte avec un texte comme message, un ou plusieurs boutons et éventuellement une icône.

Syntaxe : `var2=MsgBox("Le message", "Titre de la fenêtre")`

Exemple : `msg=MsgBox("Bonjour","Fenêtre de test")`

# IV. ENTEES ET SORTIES – OBJET FEUILLE

- MsgBox : syntaxe, arguments et constantes

La MsgBox peut être mise en œuvre de 2 manières différentes: en tant que méthode (vu ci-dessus) ou en tant que fonction

## **Exemple de MsgBox en tant que fonction:**

```
Rep=MsgBox("Etes-vous un artisan?")
```

```
If Rep = vbYes Then
```

```
    TVA=0,055
```

```
Else
```

```
    TVA=0,02
```

```
End If
```

## **Syntaxe**

```
MsgBox(prompt[, buttons] [, title] [, helpfile, context])
```

# IV. ENTEES ET SORTIES – OBJET FEUILLE

Arguments	Descriptions
<b>prompt</b>	<ul style="list-style-type: none"><li>•Chaîne de caractères représentant le message de la boîte de dialogue. La longueur maximale de cet argument est approximativement de 1024 caractères selon la police utilisée. Le passage à la ligne s'opère en insérant un retour chariot <b>vbCr</b> ou un saut de ligne <b>vbLf</b> ou bien une combinaison des deux <b>vbCrLf</b>.</li><li>Pour rappel, les constantes vbCr, vbLf et vbCrLf représentent respectivement les valeurs suivantes : Chr(13), Chr(10) et Chr(13) + Chr(10).</li></ul>
<b>buttons</b>	<ul style="list-style-type: none"><li>•Facultatif. C'est grâce à cet argument que l'on va pouvoir définir les boutons souhaités, le style d'icône et le bouton par défaut (celui qui sera pris en compte si l'utilisateur appuie sur Entrée). Il s'agit d'une donnée numérique représentant la somme des constantes possibles (voir diapo suivante). La valeur par défaut est vbOkOnly (soit la valeur 0).</li></ul>
<b>title</b>	<ul style="list-style-type: none"><li>•Facultatif. Chaîne de caractères valant titre de la boîte de dialogue. A défaut, c'est le nom de l'application qui sera pris en compte.</li></ul>
<b>helpfile *</b>	<ul style="list-style-type: none"><li>•Facultatif. Expression de chaîne indiquant le fichier d'aide à utiliser pour la boîte de dialogue. Cet argument fonctionne de paire avec l'argument <b>context</b>.</li></ul>
<b>context *</b>	<ul style="list-style-type: none"><li>•Facultatif. Expression indiquant le numéro de la rubrique d'aide associée. Cet argument fonctionne de paire avec l'argument <b>helpfile</b>.</li></ul>

Remarque : si on veut omettre certains arguments dans la définition de la MsgBox, on doit quand même placer la virgule de séparation correspondante.

L'argument **buttons** peut recevoir les valeurs (cumulables) suivantes :

Constantes	Valeurs	Descriptions
<b>vbOKOnly</b>	0	• Affiche le bouton  uniquement.
<b>vbOKCancel</b>	1	• Affiche les boutons  
<b>vbAbortRetryIgnore</b>	2	• Affiche les boutons   
<b>vbYesNoCancel</b>	3	• Affiche les boutons   
<b>vbYesNo</b>	4	• Affiche les boutons  
<b>vbRetryCancel</b>	5	• Affiche les boutons  
<b>vbCritical</b>	16	•  Affiche l'icône Message critique
<b>vbQuestion</b>	32	•  Affiche l'icône Question
<b>vbExclamation</b>	48	•  Affiche l'icône Point d'exclamation
<b>vbInformation</b>	64	•  Affiche l'icône Information
<b>vbDefaultButton1</b>	0	• Le premier bouton est le bouton par défaut.
<b>vbDefaultButton2</b>	256	• Le deuxième bouton est le bouton par défaut.
<b>vbDefaultButton3</b>	512	• Le troisième bouton est le bouton par défaut.
<b>vbDefaultButton4</b>	768	• Le quatrième bouton est le bouton par défaut.
<b>vbApplicationModal</b>	0	• Boîte de dialogue modale. L'utilisateur doit répondre au message affiché dans la zone de message avant de pouvoir continuer de travailler dans l'application en cours.
<b>vbSystemModal</b>	4096	• Modal système. Toutes les applications sont interrompues jusqu'à ce que l'utilisateur réponde au message affiché dans la zone de message.
<b>vbMsgBoxHelpButton</b>	16384	• Ajoute le bouton Aide à la zone de message.
<b>VbMsgBoxSetForeground</b>	65536	• Indique la fenêtre de zone de message comme fenêtre de premier plan.
<b>vbMsgBoxRight</b>	524288	• Le texte est aligné à droite.
<b>vbMsgBoxRtlReading</b>	1048576	• Indique que le texte doit apparaître de droite à gauche sur les systèmes hébraïques et arabes.

Le premier groupe de valeurs (0 à 5) : décrit le nombre et le type de boutons de la boîte de dialogue.

Le deuxième groupe (16, 32, 48 et 64) : décrit le style d'icône.

Le troisième groupe (0, 256 et 512) : définit le bouton par défaut.

Enfin, le quatrième groupe (0 et 4096) : détermine la modalité de la zone de message (*non utilisable en VBA*).



## IV. ENTEES ET SORTIES – OBJET FEUILLE

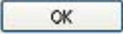



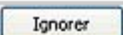


### Exemple:

Si vous voulez obtenir une MsgBox avec les boutons *Oui/Non* (valeur 4), une icône *Question* (valeur 32) et le deuxième bouton par *défaut* (valeur 256), il conviendra de saisir 292 comme argument ***buttons*** (soit  $4 + 32 + 256$ ).


Mais au lieu de saisir la valeur 292, on peut aussi saisir simplement l'expression *vbYesNo + vbQuestion + vbDefaultButton2*.

# IV. ENTEES ET SORTIES – OBJET FEUILLE

## Valeur de retour

Si l'utilisateur clique sur...	...voici la valeur retournée...	...et voilà la constante VBA correspondante.
	1	<code>vbOK</code>
	2	<code>vbCancel</code>
	3	<code>vbAbort</code>
	4	<code>vbRetry</code>
	5	<code>vbIgnore</code>
	6	<code>vbYes</code>
	7	<code>vbNo</code>

### Remarques

La croix de fermeture de fenêtre (rouge) est grisée et ne peut être sélectionnée lorsque la boîte de dialogue comprend plusieurs boutons sans la présence d'un bouton . Il n'existe donc pas de valeur de retour nulle.

Si la boîte de dialogue est dotée d'un bouton , appuyer sur **Échap** équivaut à cliquer sur .

## 3. Objet feuille de calcul

- La feuille est un objet (Cf cours suivant)

La feuille 1 du classeur actif est désignée par:

*ActiveWorkbook.Sheets("Feuil1")*

- La feuille est constituée de plusieurs cellules. Celles-ci sont repérées par un couple de deux entiers, représentant respectivement le numéro de ligne, et le numéro de colonne :

Feuil1.Cells(i,j) désigne la cellule de coordonnées (i,j) dans la feuille de nom Feuil1.

exemple:

Si l'on veut faire référence à la cellule B1 de la feuille 1 du classeur actif:

*ActiveWorkbook.Sheets("Feuil1").Cells(1,2)*