Introduction au module : exercices corrigés en VBA
Corrigé

Objectifs

- Installer l'environnement de développement ;
- Se familiariser avec l'environnement de développement ;
- Choisir les bons types ;
- Écrire un premier programme ;
- Manipuler les entrées/sorties ;

Exercice 1 : Affichage d'une ligne d'une facture *6*

1 Utilisation de l'environnement Visual Basic sous Excel

Le langage Visual Basic est l'une des nombreuses variantes du très ancien langage BASIC (1964), initialement conçu pour faciliter l'apprentissage de la programmation pour des non-spécialistes (BASIC = Beginner's All purpose Symbolic Instruction Code = langage de programmation à usage général pour débutants). Le Visual Basic est une adaptation développée par Microsoft qui a considérablement fait évoluer le BASIC des origines en lui apportant les concepts de l'algorithmique moderne. Il peut être utilisé à partir de différents environnements :

- Soit à partir de la plate-forme Visual Basic permettant de créer des applications autonomes (dernières versions : VB 6.0 puis VB.NET)
- Soit à partir d'applications dont la finalité première n'est pas la programmation, mais qui peuvent être enrichies de fonctionnalités supplémentaires programmées dans une version Visual Basic intégrée : **le VBA = Visual Basic édition Application** ; c'est le cas des applications de la famille Microsoft Office telles qu'Excel, Word, Access ou Powerpoint, ou encore d'applications plus spécialisées (par exemple le logiciel d'information géographique ArcGIS) ; le VBA est un sous-ensemble du Visual Basic, dont la limite principale est qu'il ne peut pas créer de programmes autonomes susceptibles de fonctionner indépendamment de l'application qui l'héberge ; il permet en revanche de pouvoir utiliser totalement tous les objets spécifiques de l'application-hôte, par exemple les feuilles de calcul et les graphiques dans Excel.

Tous les exemples qui seront traités dans le cadre du présent module d'algorithmique de base pourront s'exécuter dans n'importe quel environnement VB ou VBA. Nous avons privilégié l'utilisation de l'environnement Excel, considérant sa très large popularité et sa finalité propre d'outil de calcul.

1.1 Terminologie : procédure, module, projet, éditeur Visual Basic

En Visual Basic, une unité de programme s'appelle une **procédure**, ou une **macro** (nous verrons plus tard les nuances entre ces 2 termes). Une procédure est une liste d'instructions, en nombre généralement restreint, respectant la syntaxe du langage Visual Basic, et auxquelles on a donné un nom.

La saisie ou la modification d'une procédure s'effectue au sein d'une application appelée **Editeur Visual Basic**, qui s'exécute sous forme d'application séparée d'Excel (dans une fenêtre Windows différente), mais qui ne peut se lancer que depuis Excel et se referme automatiquement si on ferme Excel. Les procédures sont écrites dans des **modules**, qui peuvent être vus comme des " feuilles de programme " du classeur Excel courant, non visibles par l'application Excel mais uniquement par l'éditeur Visual Basic.

Un programme complet, associé à un classeur Excel, est appelé **projet** : il est généralement constitué de plusieurs **modules**, chacun d'entre eux regroupant plusieurs procédures.

1.2 Chargement de programmes

Les projets Visual Basic créés sous Excel sont sauvegardés dans un fichier-classeur d'extension XLS. A l'ouverture d'un tel classeur (par exemple S0-cours.xls), la présence de modules en son sein peut donner lieu à divers comportements selon le paramétrage du niveau de sécurité d'Excel :

- **Sécurité élevée** : les macros du classeur sont automatiquement désactivées (à moins que le classeur ait fait l'objet d'une " signature numérique " en vue de sa diffusion)
- **Sécurité moyenne** : une fenêtre d'alerte-macro s'affiche invitant l'utilisateur à activer ou désactiver les macros ; en présence d'un classeur Excel dont vous ne connaissez pas l'origine, le réflexe doit être de désactiver les macros, qui sont généralement des " macro-virus " susceptibles de perturber ou endommager le fonctionnement de l'ordinateur ! En revanche, pour tous les classeurs que nous échangerons pendant la présente formation d'algorithmique, il faudra bien évidemment activer les macros afin de pouvoir exécuter et tester les programmes écrits.
- **Sécurité faible** : les macros du classeur sont automatiquement activées ; en particulier, si une macro est attachée à l'évènement " ouverture du classeur ", son exécution est lancée (c'est ainsi que procèdent tous les virus de macro) : il est évident qu'il ne faut jamais configurer Excel (ou toute autre application) sur un niveau faible de sécurité !

Remarque :

Pour plus de détails sur la sécurité, voir l'aide en ligne d'Excel (menu " ? / Aide sur Microsoft Excel " ou touche F1), onglet Index, mot-clé " Macro ", rubrique " A propos de la sécurité des macros ".

Par défaut, Excel est configuré avec un niveau de sécurité élevé, qui ne nous permet pas d'exécuter les programmes que nous créerons. **Il faut donc passer en niveau moyen de sécurité** : valider le menu " Outils / Macro / Sécurité... ", onglet " Niveau de sécurité ", et cocher " Moyenne ".

Attention :

Il faut quitter Excel et le réouvrir pour que ce nouveau réglage prenne effet.

1.3 Exécution de programmes depuis Excel (Alt-F8)

Depuis Excel, valider le menu " **Outils / Macro / Macros ...** " **(ou raccourci Alt-F8)** : la liste des procédures exécutables s'affiche dans une fenêtre pop-up intitulée " Macro ". Double-cliquer sur la procédure choisie (par exemple pgcd) pour lancer son exécution.

Remarque :

On peut également créer un bouton sur la feuille de calcul et lui associer la procédure de telle sorte que tout clic sur le bouton lance l'exécution de la procédure : pour réaliser cela, afficher la barre d'outils " Formulaires " (menu " Affichage / Barres d'outils / Formulaires "), cliquer sur l'icône en forme de bouton dans cette barre d'outil, puis dessiner le contour rectangulaire du bouton par clic et " glisser " de la souris ; au relâchement, la fenêtre " Affecter une macro " s'affiche : double-cliquer sur le nom de la procédure voulue pour l'associer au bouton. Il reste à modifier le libellé affiché sur le bouton en cliquant sur le centre du bouton encore sélectionné pour taper le texte voulu. Ces 2 opérations (affecter une macro ou modifier le texte du bouton) peuvent être réalisées a posteriori en cliquant droit sur le bouton (à condition qu'il ne soit pas en état " sélectionné " à ce moment : si c'est le cas, cliquer hors du bouton pour le désélectionner).

1.4 Consultation de programmes : éditeur Visual BASIC (Alt-F11)

Pour consulter le code Visual Basic du programme précédent, réouvrir la fenêtre " Macro " (utiliser dorénavant le raccourci Alt-F8), sélectionner par simple clic la procédure pgcd, puis cliquer sur le bouton " Modifier " : ceci provoque le lancement de l'**éditeur Visual Basic** dans une fenêtre d'application nouvelle, incluant une sous-fenêtre ouverte sur le module contenant la procédure pgcd, avec le curseur positionné en début de cette procédure.

Il est également possible de lancer directement l'éditeur Visual Basic depuis Excel avec le menu " **Outil / Macro / Visual Basic Editor** " **(raccourci Alt-F11)**.

Une fois sous l'éditeur Visual Basic, il est utile de faire apparaître le volet **Explorateur de projet** par le menu " **Affichage / Explorateur de projet** " **(ou raccourci Ctrl-R)** : ceci permet de lister tous les projets actuellement ouverts (avec au minimum le classeur courant S0-cours.xls) et de montrer la structure de chacun :

- Catégorie " Microsoft Excel Objets " : elle indique une entrée pour chaque feuille de calcul du classeur, ainsi qu'une autre pour le classeur lui-même (ThisWorkbook) ; dans l'immédiat, nous n'utiliserons pas ces entrées, destinées à des usages plus approfondis où l'on souhaite attacher des procédures à des évènements particuliers touchant le classeur ou une feuille de calcul
- Catégorie " Modules " : elle indique la liste des modules constituant le projet ; pour consulter le code d'un module, il faut double-cliquer sur son nom

1.5 Sauvegarde de programmes (Ctrl-S)

La sauvegarde d'un projet, c'est-à-dire de l'ensemble des modules qui le constituent, s'effectue en enregistrant le classeur correspondant, soit depuis Excel (" Fichier / Enregistrer " ou bouton " Enregistrer " ou raccourci Ctrl-S), soit depuis l'éditeur Visual Basic (mêmes options).

Il est également possible de sauvegarder isolément un module sous forme de fichier d'extension .BAS en cliquant droit sur son nom dans la fenêtre " Explorateur de projet " et en validant l'option " Exporter un fichier ". Ceci est utile pour pouvoir récupérer ce module dans d'autres environnements Visual Basic (VBA pour Word, par exemple). Symétriquement, un module peut-être chargé à partir d'un fichier .BAS en cliquant droit sur le projet dans la fenêtre " Explorateur de projet " et en validant l'option " Importer un fichier ".

1.6 Création de programmes et exécution depuis l'éditeur Visual Basic (F5)

Dans l'éditeur Visual Basic, il faut rajouter un nouveau module en validant le menu " Insertion / Module ". On saisit ensuite le texte de la procédure entre les lignes **`Sub`** `NomDuProgramme` et **`End Sub`**. Puis on sauvegarde le projet sur disque avec le menu " Fichier / Enregistrer " - ou Ctrl-S. A titre d'exemple, insérer un nouveau module (il prendra le nom par défaut " Module1 ") et taper le programme suivant :

```
1
2 Sub AfficherBonjour ()
3
4     MsgBox "Bonjour"
5
6 End Sub
```

Remarque :

Pour renommer un module, il faut afficher la fenêtre " Propriétés " (menu " Affichage / Fenêtre Propriétés " ou raccourci F4) et saisir le nom voulu au droit de la propriété **`(Name)`** (par exemple ici : Test). On peut ensuite refermer cette fenêtre en cliquant sur le bouton de fermeture (symbole X).

On peut tester immédiatement ce programme depuis l'éditeur Visual Basic en lançant son exécution : positionner le curseur n'importe où dans la procédure puis :

- valider l'option " Exécuter la macro " du menu Exécution (ou **raccourci F5**)
- ou cliquer sur le bouton " Exécuter la macro " de la barre d'outils standard de l'éditeur Visual Basic

Si le curseur n'est pas positionné dans une procédure, une boîte de dialogue affiche la liste des procédures définies dans tous les modules des classeurs ouverts

1.7 Débogage : exécution " pas à pas " (F8, Maj-F8), points d'arrêt (F9)

Il est possible d'exécuter le programme dans le **mode " pas à pas "** qui provoque une suspension de l'exécution après chaque instruction traitée :

– valider l'option " Pas à pas détaillé " du menu " Débogage "
– ou cliquer sur le bouton " Pas à pas détaillé " de la barre d'outils " Débogage "
– ou appuyer le **raccourci F8**

La fenêtre du module contenant le code en cours d'exécution s'affiche alors, la prochaine instruction à exécuter étant surlignée en jaune, en l'occurrence ici la ligne `Sub PremierProgramme`. On peut ensuite réitérer la même opération avec le **raccourci Maj-F8** (maintien de la touche majuscule puis appui de F8), correspondant à l'option " Pas à pas principal ", pour exécuter une instruction de plus.

Remarque :

La nuance entre les raccourcis F8 et Maj-F8 sera précisée ultérieurement lors de l'étude des procédures et fonctions.

A tout moment, on peut également poursuivre l'exécution normale du programme jusqu'à la fin :

– valider l'option **" Continuer "** du menu Exécution
– ou cliquer sur le bouton " Continuer " (barre " Débogage ")
– ou appuyer sur le **raccourci F5**

On peut enfin clore l'exécution du programme définitivement (en particulier si on veut ensuite relancer l'exécution depuis le début) :

– valider l'option **" Réinitialiser "** du menu " Exécution "
– ou cliquer sur le bouton " Réinitialiser " (barre " Débogage ")

Attention :

Cette opération sera notamment à réaliser toute les fois où un programme se sera bloqué sur une condition d'erreur ; pour relancer le programme après correction de l'erreur, il sera indispensable de le réinitialiser, sans quoi on aura une fenêtre d'erreur qui indiquera : " impossible d'exécuter le programme en mode Arrêt ".

Il est également possible d'insérer des " **points d'arrêt** " dans un programme en cliquant dans la marge de la ligne voulue (zone grise) ou en validant l'option " Basculer le point d'arrêt " du menu " Débogage ", ou en cliquant sur le bouton " Basculer le point d'arrêt " de la barre " Débogage ", ou encore en frappant le **raccourci F9**. Lorsque l'exécution du programme passe sur un point d'arrêt, l'exécution est stoppée et peut ensuite être relancée manuellement selon les méthodes précédemment vues.

On supprime un point d'arrêt de la même manière qu'on l'a inséré (" Basculer le point d'arrêt ").

Le curseur peut aussi être utilisé comme point d'arrêt temporaire en validant l'option " Exécuter jusqu'au curseur " du menu " Débogage ", ou en frappant le raccourci Ctrl-F8.

D'autres facilités de débogage existent et seront présentées ultérieurement (points d'arrêt conditionnel, fenêtre " Espions ", fenêtre " Variables locales ", ...etc.).

1.8 Configuration de l'éditeur Visual Basic

Les paramétrages principaux de l'environnement Visual Basic s'obtiennent par le menu " Outils / Options ". Nous utiliserons les réglages par défaut à l'exception d'un seul, qui nous apportera une sécurité dans l'utilisation des variables : dans l'onglet " Editeur ", au sein de la

zone " Paramètres du code ", **cocher la case " Déclaration des variables obligatoire "**. Ce réglage demeurera valide pour toutes les sessions ultérieures de l'éditeur Visual Basic et aura pour effet de rajouter automatiquement une première ligne Option Explicit lors de la création de tout module. Il permettra notamment de nous garantir contre un risque d'erreur classique : la mauvaise orthographe d'un nom de variable.

1.9 Séparateur décimal

Le séparateur décimal pour les nombres réels peut correspondre au point ou à la virgule selon la configuration de votre système d'exploitation. Sous Windows, ce réglage peut s'effectuer via le Panneau de configuration, Options régionales et linguistiques, onglet " Options régionales ", bouton " Personnaliser ", zone " Symbole décimal ". Dans le cadre de cet enseignement, nous utiliserons le point décimal (convention internationale).

2 Exercice

Exercice 1 : Affichage d'une ligne d'une facture
Écrire un programme pour saisir le code d'un article (un seul caractère), un prix unitaire hors taxe (exprimé en euros), une quantité entière. Il affiche ensuite un récapitulatif en donnant en plus le prix total hors taxes et TTC.

Par exemple, l'article de code T et de prix unitaire 2,5 a été commandé en 100 exemplaires. La ligne de facture affichée est donc la suivante :

```
T              2.50      100            250.00            299.00
```

Solution :

```
1  R0 : Afficher une ligne d'une facture
2
3  tests : celui fournit dans le sujet.
4
5  R1 : Raffinage de « Afficher une ligne d'une facture »
6    | Saisir les caractéristiques de l'article commandé
7    | Calculer les prix
8    | Afficher la ligne de la facture
9
10 R2 : Raffinage De « Saisir les caractéristiques de l'article commandé »
11   | Saisir le code
12   | Saisir le prix unitaire
13   | Saisir la quantité
14
15 R2 : Raffinage De « Calculer le prix »
16   | prix_ht <- quantité * prix_unitaire
17   | prix_ttc <- prix_ht * (1 + TVA)
18
19 R2 : Raffinage De « Afficher la ligne de la facture »
20   | Écrire(code, prix_unitaire, quantité, prix_ht, prix_ttc)
```

On en déduit alors l'algorithme suivant :

```
Algorithme ligne_facture

        -- Afficher une ligne d'une facture

Constante
    TVA = 0.196         -- Taux de TVA

Variable
    code: Caractère     -- le code de l'article
    prix_unitaire: Réel -- le prix unitaire
    quantité: Entier    -- quantité de l'article
    prix_ht: Réel       -- prix hors taxes
    prix_ttc: Réel      -- prix TTC

Début
    -- Saisir les caractéristiques de l'article commandé
    Saisir le code
    Saisir le prix unitaire
    Saisir la quantité

    -- Calculer les prix
    prix_ht <- quantité * prix_unitaire
    prix_ttc <- prix_ht * (1 + TVA)

    -- Afficher la ligne de la facture
    Écrire(code, prix_unitaire, quantité, prix_ht, prix_ttc)
Fin
```

```
Attribute VB_Name = "io_quantite_produit"
'*****************************************************************
'*  Auteur   : Claude Monteil <monteil@ensat.fr>
'*  Version  : 1.0
'*  Objectif : Afficher une ligne d'une facture (sans controle)
'*****************************************************************

Sub ligne_facture()
    Const TVA As Double = 0.196 ' Taux de TVA
    Dim code As String          ' le code de l'article
    Dim prix_unitaire As Double ' le prix unitaire
    Dim quantite As Integer     ' quantite de l'article
    Dim prix_ht As Double       ' prix hors taxes
    Dim prix_ttc As Double      ' prix TTC

   '1.Saisir les caracteristiques de la ligne de facture
    Afficher "Code article :"
    Saisir code
    Afficher "Prix unitaire :"
    Saisir prix_unitaire
    Afficher "Quantite :"
    Saisir quantite

```

```
24      '2.Calculer les prix
25      prix_ht = quantite * prix_unitaire
26      prix_ttc = prix_ht * (1 + TVA)
27
28    '3.Afficher la ligne de la facture
29      Afficher code, prix_unitaire, quantite, prix_ht, prix_ttc
30  End Sub
```