

Langages d'automatisation : VBA

Sacha Kozlov
(green.fr@gmail.com)

EFREI
2011

Introduction

- Présentation
- But du cours
- Pourquoi VBA ?
- Pré-requis
- Notations

Présentation

Sacha Kozlov – green.fr@gmail.com

Délégué ? Groupe de diffusion ? Forum ?

But du cours

- Pragmatisme : ne pas savoir utiliser les logiciels bureautiques – être illettré aujourd'hui
- Périmètre : VBA de MS Excel
 - VBA (comparaison avec java)
 - Les classes de MS Excel (introduction rapide)

Pourquoi VBA ?

- A chaque besoin – son langage
- VBA – automatisation des tâches
- Couvre d'autres produits (MS Office, Open Office, Lotus Notes, etc...)
- Le principe est similaire pour d'autres langages
- Bien documenté (<http://www.excelfunctions.net>)

Les inconvénients du VBA

- Le groupe cible du VBA est trop large
 - Langage est trop laxiste (gestion « par défaut »)
 - Rareté de la littérature pour les informaticiens
 - Format binaire du stockage (pas de diff, etc.)
 - Les dernières versions sont au format ZIP/text
 - Les macros sont au format binaire (OLE)

Les pré-requis

- Connaissance de java
- Utilisation sommaire d'Excel
- Anglais informatique

Notations

- Tout texte en Courier est le code :
 - `Option Explicit`
 - `Dim i as Integer`

Éléments du langage

- Organisation du code
- Présentation du code
- Les variables
- Opérateurs et fonctions
- Contrôle d'exécution
- Méthodes
- Gestion des erreurs

Organisation du code

- Module – unité de code
 - Module standard
 - Procédures, fonctions
 - Variables globales
 - Options
 - Module de classe (hors périmètre)
 - Module `UserForm` (hors périmètre)
- Ambiguïté du terme « macro »

Présentation de IDE

- Explorateur du projet
- Editeur du code
- Fenêtre d'exécution et les espions
 - Possibilité de modification des variables
- Le éléments est cassiques de IDE

Organisation du code – avancé

- Appel avec les raccourcis clavier
- Menu et barre d'outil d'utilisateur
- Fichiers XLA
 - « perso.xla »
 - Macros complémentaires

Présentation du code

- Convention des noms – java
 - Sauf les mots-clés de VBA et d'Excel
 - Case-insensitive
- Alignement du code – obligatoire
- Commentaires – compromis intelligent
- Sauts des lignes trop longues

Les variables – type de données

- Pas de classes dans les types standard
 - Types numériques : Integer, Double, etc...
 - Type String
 - Type Date
- Type des données générique – Variant

Les variables – déclaration

- Option explicit
 - Obligation de déclarer toutes les variables
 - Sinon – la déclaration implicite en Variant
- Déclaration avec Dim
 - Dim counter As Integer
 - « bug » avec la déclaration de plusieurs variables
 - Dim x, y As Double

Les variables – attribution

- Impossible d'affecter une valeur en même temps que la déclaration de la variable
 - Java : `int x = 12 ;`
 - VBA :
 - `Dim x As Integer`
 - `X = 12`
- Let **et** Set – rudiments
 - `Let x = 12` est équivalent à `x = 12`
 - `Set a = b` – mot-clé obligatoire pour les objets

Les variables – la portée

- Variables globales
 - Définition dans la zone « Déclarations »
 - `Public` – visibles partout dans le projet
 - `Private` – visibles uniquement dans le module
- A l'intérieur des méthodes – comme java
 - La portée restreinte par la méthode
 - Les variables définies dans un bloc (`If`, `For`, etc) ne sont visible qu'à l'intérieur du bloc

Constantes

- Déclaration avec `Const`
 - `Public Const DAYS_WEEK As Integer = 7`
 - L'initialisation au même moment que la déclaration
- Peuvent être déclarées en globales
 - Publiques ou privées
- Ou à l'intérieur des fonctions
 - La portée comme des variables classiques

Les variables – Array fixe

- Déclaration des tableaux fixes avec `Dim` :
 - Le tableau unidimensionnel
 - `Dim a(0 To 9) as Integer`
 - Le tableau multidimensionnel :
 - `Dim a(1 To 10, 11 To 20) as Integer`
 - Le syntaxe déconseillé (uniquement la taille) :
 - `Dim a(10, 10) as Integer`
- Option `Base` pour spécifier le début des indices par défaut

Les variables – Array dynamic

- Déclaration des tableaux variables :
 - Etape de déclaration sans préciser la taille
 - `Dim a() as Integer`
 - Etape de dimensionnement
 - `ReDim a(10)`
 - La possibilité de garder les anciennes valeurs
 - `ReDim Preserve a(30)`

Opérateurs et fonctions

- Opérateurs et fonctions arithmétiques
- Opérateurs booléens
- Manipulations des chaînes
- Manipulations des dates
- Manipulation de type des variables

Opérateurs arithmétiques

- Literals – à l'américaine
 - 120, 3.1415926535, etc.
- Les opérateurs classiques :
 - « + », « - », « * », « / », « ^ »
- Les opérateurs de la division des entiers :
 - « \ », « Mod »
- Les opérateurs de comparaison :
 - « = », « < », « > », « <= », « >= », « <> »

Fonctions arithmétiques

- Les fonctions classiques :
 - Round, Exp, Ln, Rand, Sin, Cos, etc...
- Les fonctions d'agrégation :
 - Sum, Product, etc...
- Absence des fonctions `Min` et `Max`
 - Accessibles via les classes Excel

Opérateurs booléens

- Literals – « True », « False »
 - L'affichage est traduit en français
- Les opérateurs classiques :
 - And, Or, Not, etc...
- Comparaison possible par le changement implicite des types :
 - $\text{True} \rightarrow -1$
 - $\text{False} \rightarrow 0$

Opérateurs des chaînes

- Literals – entre guillemets
 - Problème d'échappement – double guillemets
 - "Les ""bons"" élèves" → Les "bons" élèves
- Rudiment de « \$ » dans les noms
- Opérateur « & » pour la concaténation
 - Une fonction `Concatenate` existe également

Fonctions des chaînes

- Fonctions classiques avec des noms MS :
 - `Left`, `Right`, `Mid` pour découper manuellement
 - `Len` pour calculer la longueur
 - `InStr` et `Replace` pour chercher et remplacer
- Fonctions utiles :
 - `Split` pour découper la chaîne avec un séparateur

Opérateurs des dates

- Literals – à l'américaine, avec des « # » :
 - #10/12/2011# pour 12 octobre 2011
- Format interne : numérique :
 - Unité = 1 jour
 - Gestion des années à 2 chiffres – pragmatisme !

Fonctions des dates

- Fonctions de découpage :
 - `Day`, `Month`, `Year`, `Hour`, `Weekday`, etc...
- Format très puissant avec `Format`
- Calculs pratiques :
 - `DateDiff` avec des différentes conventions
 - `DateAdd` pour ajouter 1 mois, etc. à la date

Manipulation des types

- Abondance des changements de type implicites
 - Souvent très dangereux, illisible, à éviter
- Fonctions « `Is . . .` » :
 - `IsDate`, `IsNumeric`, etc. pour trouver le type
 - `IsError`, `IsMissing`, etc. pour contexte
- Changement de type : fonctions « `C . . .` » :
 - `CDate`, `CDouble`, etc...

Contrôle d'exécution – If

- La structure classique :

- If ... Then ... Else ... End If

- Le cas spécial du ElseIf

```
If (x < 0.25) Then
    Debug.Print "Premier quadrile"
ElseIf (x < 0.5) Then
    Debug.Print "Deuxième quadrile"
Else
    Debug.Print "Après la médiane"
End If
```

Contrôle d'exécution – Select

- `Select Case ... Case ... End Select`
 - `Case Else`
 - `Case Is`

```
Select Case y
    Case 1, 3, 5:
        Debug.Print "Impair"
    Case Else
        Debug.Print "Pair"
End Select
```

Autres instructions de bifurcation

- Opérateur `GoTo` – illisible, à éviter
 - Introduit des libellés qui sont utilisées dans la gestion des erreurs

- Fonction « `IIf` » :

```
IIf(y < 4, "Petit nombre", "Grand nombre")
```

Contrôle d'exécution – boucle For

- `For ... Next` pour une boucle for classique
 - La version pour les collections – voir la suite
- `Exit For` pour forcer la sortie

```
For i = 1 To 50
    x = Rnd
    If (x > 0.99) Then
        Exit For
    End If
Next
```

Boucles conditionnelles

- **Boucles pré-conditionnelles**
 - `Do While ... Loop` **et** `Do Until ... Loop`
- **Boucles post-conditionnelles**
 - `Do ... Loop While` **et** `Do ... Loop Until`
- **Exit Do pour forcer la sortie**

```
Do While (x < 0.99)
    x = Rnd
Loop
```

Méthodes – I

- Fonctions et procédures
 - La fonction peut mais n'est pas obligée de renvoyer une valeur
 - Redondance, on va utiliser les fonctions
- Bug avec « Call » si plus d'un paramètre :
 - MyFun (10) – OK
 - A = MyFun (10, 20) – OK
 - MyFun (10, 20) – KO
 - Call MyFun (10, 20) – OK

Méthodes – II

- Paramètres typés :
 - `Function doIt(x As Double, y As Double)`
- `Optional` pour les paramètres optionnels
- `ParamArray` pour une liste des paramètres
- Paramètres nominatifs
- `ByVal` pour passer par valeur (protection)
- `ByRef` pour passer par référence (vitesse, changement)

Méthodes – III

- Une seule valeur de retour
 - On peut contourner avec des paramètres `ByRef`
- Variable de retour – homonyme de la fonction
 - La variable est créée automatiquement
 - On peut l'utiliser pour les calculs
- Forçage de sortie avec `Exit Function`

Exercice : fonction min2

- Ecrire une fonction qui recherche la deuxième plus petite valeur parmi les valeurs passées
- Les valeurs sont passées sous forme d'une variable Variant contenant une matrice
- Adapter la fonction pour les appels depuis une feuille Excel

Hints pour l'exercice

- Pour simplifier les tests, utiliser `Array`

```
minimum2 (Array (1, -2, -5, 9, -5))
```

- Comme conséquence, la déclaration

```
Function minimum2 (data As Variant) As Integer
```

- Fonctions `LBound` et `UBound`

Développement d'exercice

- Adapter la fonction à l'appel depuis Excel :
 - Le paramètre devient de type `Range`
 - La propriété `.Value` donne un `Variant` avec un tableau 2D des valeurs sélectionnées
 - Il est préférable de faire deux fonctions :
 - Une fonction VBA pure
 - Une fonction « proxy » avec Excel