

# Initiation à Visual Basic pour Applications

## Examen (1h)

Kévin Huguenin<sup>1</sup> et Romain Tavenard<sup>2</sup>

Examen : correction

14 Novembre 2007

### 1. Syntaxe de VBA

- a. Déclaration de variables : laquelle de ces déclaration est une déclaration correcte d'une variable `intAge` de type `Integer` en VBA ?

(a) `dim Integer intAge`                      (b) `Integer intAge`  
(c) `dim Integer as intAge`                    (d) `dim intAge as Integer`

La bonne réponse était `dim intAge as Integer`.

- b. Déclaration de constante : la déclaration suivante est-elle correcte ?

```
const strName as String
```

(a) Oui, les `String` n'ont pas à être initialisées      (b) Non, il faut initialiser toutes les constantes  
(c) Oui, si on définit `strName` plus tard dans le code (avant de l'utiliser)      (d) Non, on ne peut pas définir de constante de type `String`

La bonne réponse était « Non, il faut initialiser toutes les constantes ».

- c. Tableaux : quel est l'effet de la déclaration suivante ?

```
Dim t(3 To 10) as Integer
```

(a) Déclare un tableau `t` dont la taille peut varier entre 3 et 10      (b) Déclare un tableau `t` de taille 10 dont les indices commencent à 3  
(c) Déclare un tableau `t` dont les indices varient de 3 à 10      (d) Déclare un tableau `t` avec deux cases d'indices 3 et 10

La bonne réponse était « Déclare un tableau `t` dont les indices varient de 3 à 10 ».

- d. Tableaux (suite) : que vaut `t(0)` après les opérations suivantes ?

```
Dim t() as Integer
Dim i As Integer
ReDim t(10)
```

<sup>1</sup> Kevin.Huguenin@eleves.bretagne.ens-cachan.fr

<sup>2</sup> Romain.Tavenard@irisa.fr

```

For i=LBound(t) To UBound(t)
    t(i) = i + 1
Next i
ReDim t(20)

```

- (a) les indices du tableau commencent à 1  
 (b)  $t(0) = i + 1$   
 (c)  $t(0) = 0$   
 (d)  $t(0) = 1$

La bonne réponse était  $t(0) = 0$ . En effet, la boucle affecte bien la valeur 1 à  $t(0)$ , mais lorsque l'on redimensionne  $t$  sans utiliser le mot-clé `Preserve`, les données sont réinitialisées à 0.

- e. Fonctions : la fonction suivante est-elle syntaxiquement correcte ?

```

Public function factorielle (Byval n as Integer) as Long
    factorielle(0) = 1
    If n>0 then
        factorielle(n) = n*factorielle(n-1)
    End If
End Function

```

- (a) Oui, tout va bien  
 (b) Non, l'affectation de la valeur de retour est incorrecte  
 (c) Non, une fonction qui prend en argument un Integer ne peut pas renvoyer un Long  
 (d) Non, il manque le sous-bloc Else dans le bloc If

La bonne réponse était « Non, l'affectation de la valeur de retour est incorrecte ». En effet, une fonction est appelée par `nomDeLaFonction(arguments)`, mais la valeur de retour est affectée par `nomDeLaFonction = valeurDeRetour`.

## 2. Programmation VBA : généralités

- a. Affectations : quelles sont les valeurs respectives des variables  $a$  et  $b$  après l'exécution du code suivant ?

```

Dim a as Integer
Dim b as Integer
a = 1
b = 2
a = b
b = a

```

- (a)  $a=1$  et  $b=1$       (b)  $a=1$  et  $b=2$       (c)  $a=2$  et  $b=1$       (d)  $a=2$  et  $b=2$

La bonne réponse était  $a=2$  et  $b=2$ . Il suffit d'exécuter l'algorithme pas à pas pour s'en convaincre.

- b. Boucles `for` : quelle valeur initiale faut-il affecter à  $somme$  pour qu'elle calcule la somme des chiffres de 3 à  $n$  ?

```

Public function somme(Byval x as Integer, Byval n as Integer)
as Long
    Dim i as Integer
    somme = ?
    For i = 3 To n
        Somme = somme + i
    Next i
End Function

```

- (a) 0                      (b) 1                      (c) 3                      (d) n

La bonne réponse était 0, car pour calculer une somme, il faut partir de zéro et ajouter les éléments un à un.

- c. Fonctions récursives: que calcule la fonction suivante ?

```
Public function mystere(Byval x as Integer,Byval n as Integer)_
as Long
    If n = 0 Then
        mystere = 1
    Else
        mystere = x*mystere(x,n-1)
    End If
End Function
```

- (a)  $x \times n!$                       (b)  $n^x$                       (c)  $x^n$                       (d) aucun des trois

La bonne réponse était  $x^n$ , comme vu en TD.

- d. Procédures : que fait la procédure suivante ?

```
Public sub swap(Byval a as Integer, Byref b as Integer)
    Dim aux as Integer
    aux = a
    a = b
    b = aux
End sub
```

- (a) Elle échange les valeurs de a et b                      (b) Elle met la valeur de a dans b et ne change pas a  
(c) Elle met la valeur de b dans a et ne change pas b                      (d) Rien

La bonne réponse était « Elle met la valeur de a dans b et ne change pas a ». En effet, a étant passé par valeur, sa valeur est modifiée dans le corps de la fonction mais pas dans la procédure qui l'appelle. En revanche, b est passé par référence et les changements qui lui sont appliqués au sein de la procédure se répercutent en dehors de celle-ci.

### 3. Programmation VBA : les objets

- a. Création d'objet: soit monSuperObjet une instance de la classe d'objet SuperObjet. Comment réalise-t-on une construction par copie, c'est-à-dire comment initialise-t-on l'instance monSuperObjet en lui faisant prendre la valeur prise par une instance monAutreSuperObjet déjà existante ?

- (a) monSuperObjet = monAutreSuperObjet  
(b) on doit copier une à une les valeurs des propriétés de monAutreSuperObjet dans monSuperObjet  
(c) monSuperObjet.copy(monAutreSuperObjet)  
(d) aucune des trois réponses

La bonne réponse était « aucune des trois réponses ». En effet, la construction par copie se fait à l'aide du mot-clé `Set`. La ligne correspondant à l'énoncé serait donc :

```
Set monSuperObjet = monAutreSuperObjet
```

- b. Création d'objet : soit `monSuperObjet` une instance de la classe d'objet `SuperObjet`. Comment réalise-t-on une construction par création, c'est-à-dire comment initialise-t-on l'instance `monSuperObjet` en lui faisant prendre la valeur par défaut ?

- (a) `monSuperObjet = new SuperObjet`
- (b) `Dim monSuperObjet as new SuperObjet`
- (c) `Set monSuperObjet = new SuperObjet`
- (d) `monSuperObjet = SuperObjet`

La bonne réponse était « `Set monSuperObjet = new SuperObjet` ».

- c. Affectation des propriétés : soit `monSuperObjet` une instance de la classe d'objet `SuperObjet`. Comment affecte-t-on la valeur 3 à la propriété `maPropriete` de `monSuperObjet` ?

- (a) `monSuperObjet.maPropriete(3)`
- (b) `maPropriete(monSuperObjet) = 3`
- (c) `mapropriete(monSuperObjet, 3)`
- (d) `monSuperObjet.maPropriete = 3`

La bonne réponse était « `monSuperObjet.maPropriete = 3` ».

- d. Appel de méthode : soit `monSuperObjet` une instance de la classe d'objet `SuperObjet`. Comment utilise-t-on la méthode `FaireDesChoses` sur `monSuperObjet` ?

- (a) `call FaireDesChoses(monSuperObjet)`
- (b) `monSuperObjet.FaireDesChoses()`
- (c) `monSuperObjet = FaireDesChoses(monSuperObjet)`
- (d) `monSuperObjet : FaireDesChoses()`

La bonne réponse était « `monSuperObjet.FaireDesChoses()` ».

- e. Destruction d'objet : soit `monSuperObjet` une instance de la classe d'objet `SuperObjet`. Comment détruit-on `monSuperObjet` ?

- (a) `destroy(monSuperObjet)`
- (b) `monSuperObjet.destroy()`
- (c) `Set monSuperObjet = NULL`

(d) `Set monSuperObjet = nothing`

La bonne réponse était « `Set monSuperObjet = nothing` » comme vu dans la correction du TD n°4.

#### 4. Les objets d'Excel

- a. Gestion de la cellule active : parmi les 4 propositions suivantes, laquelle permet de sélectionner la cellule « A5 »

- (a) `ActiveCell = Range("A5")`  
 (b) `Selection = Range("A5")`  
 (c) `ActiveCell.CurrentRegion = Range("A5")`  
 (d) `Range("A5").Select`

La bonne réponse était « `Range("A5").Select` ».

- b. Manipulation des plages de cellules : après la série d'instruction suivante, quelle est la taille de la sélection désignée par `maSelection` ?

```
Dim maSelection as Range
Dim intTaille as Integer
Set maSelection = Range("A1:B3").Offset(1,3)
intTaille = maSelection.Columns.Count
maSelection.Resize(3,intTaille)
intTaille = intTaille - 1
```

- (a) 3 lignes, 2 colonnes                      (b) 0 ligne, 0 colonne  
 (c) 3 lignes, 1 colonne                      (d) 1 ligne, 3 colonnes

La bonne réponse était « 3 lignes, 2 colonnes ». En effet, la sélection de départ fait 3 lignes et deux colonnes, l' `Offset` n'y change rien puisqu'il n'effectue qu'un décalage, et enfin, le `Resize` est effectué avant la modification de la variable `intTaille`, qui n'a donc aucun effet sur la taille de `maSelection`. Quoi qu'il en soit, pour modifier la taille de ma sélection, il aurait fallu faire appel à une commande du type :

```
Set maSelection = maSelection.Resize(3,intTaille)
```

- c. Manipulation des plages de cellules (suite) : après la série d'instruction suivante, quelle sera la formule contenue dans la cellule A10 ?

```
Range("A1").Formula = "=B1-C$1"
Call Range("A1").AutoFill(Range("A1:A10"), xlFillCopy)
```

- (a) `"=B1-C$1"`      (b) `"=B10-C$10"`      (c) `"=B10-C$1"`      (d) aucun des trois

La bonne réponse était `"=B10-C$1"`. En effet, la formule de départ est recopiée vers le bas, ce qui, sous Excel, revient à incrémenter les numéros de ligne non précédés du symbole \$ et à laisser les autres intacts.

- d. Tracé de courbes : parmi les propositions suivantes, laquelle permet de tracer une courbe (utilisant les valeurs de la plage "A1:A10" de la feuille de calcul numéro 1) dans le premier graphique du classeur ?

(a) `Charts(1).SeriesCollection.Add(Worksheets(1).Range("A1:A10"))`

(b) `Charts(1).Range(Worksheets(1).Range("A1:A10"))`

(c) `Graphics(1).SeriesCollection.Add(Worksheets(1).Range("A1:A10"))`

(d) `Graphics(1).Range(Worksheets(1).Range("A1:A10"))`

La bonne réponse était « `Charts(1).SeriesCollection.Add(Worksheets(1).Range("A1:A10"))` ». L'objet `Graphics` n'existe pas sous Visual Basic pour Excel, pas plus que la méthode `Range` de l'objet `Charts`.

e. Gestion du classeur : parmi les propositions suivantes, laquelle n'insère pas une nouvelle feuille de calcul au classeur actif ?

(a) `ActiveWorkbook.Worksheets.Add`

(b) `Worksheets(1).Add`

(c) `Worksheets.Add`

(d) `Sheets.Add`

La bonne réponse (c'est-à-dire celle qui correspond à une syntaxe incorrecte !) était « `Worksheets(1).Add` ». Il s'agit là d'une différence fondamentale entre deux types d'objets manipulés : dans toutes les autres propositions, on a une collection de feuilles de calcul, à laquelle on ajoute une nouvelle feuille, alors que `Worksheets(1)` est une feuille de calcul, et l'objet feuille de calcul n'a pas de méthode `Add` !

## 5. Les objets de Word

a. On sélectionne le texte suivant

a brief survey on tHe Microsoft visual basic Programming language

Puis on exécute le code suivant

```
For Each aWord In Selection.Words
  If Trim(aWord.Text) = "a" Or Trim(aWord.Text) = "the" Then
    aWord.Case = wdLowerCase
  Else
    aWord.Case = wdTitleWord
  End If
Next aWord
```

Quel sera le résultat après l'exécution ?

(a) a Brief Survey On The Microsoft Visual Basic Programming Language

(b) A Brief Survey On the Microsoft Visual Basic Programming Language

(c) A Brief Survey On The Microsoft Visual Basic Programming Language

(d) a Brief Survey On the Microsoft Visual Basic Programming Language

La bonne réponse était « a Brief Survey On The Microsoft Visual Basic Programming Language ». En effet, lorsque le « a » est rencontré, le test renvoie "Vrai", ce qui n'est pas le cas lorsque le « tHe » est rencontré, car la comparaison de chaînes de caractères est sensible à la casse.

b. On sélectionne le texte suivant :

Voila un un un un texte texte avec plein plein de de  
doublons, mais une petite petite macro bien faite faite  
devrait les supprimer.  
Puis on exécute le code suivant

```
i = 1
While i < Selection.Words.Count
  If Selection.Words(i).Text = Selection.Words(i+1).Text Then
    Selection.Words(i).Delete
  Else
    i = i + 1
  End If
Wend
```

Quel sera le résultat après l'exécution ?

(a) Voila un un un un texte texte avec plein plein de de  
doublons, mais une petite petite macro bien faite faite  
devrait les supprimer.

(b) Voila un texte avec plein plein de doublons, mais une  
petite macro bien faite devrait les supprimer.

(c) Voila un texte avec plein plein de doublons, mais une  
petite macro bien faite devrait les supprimer.

(d) Voila un texte avec plein de doublons, mais une petite  
macro bien faite devrait les supprimer.

La bonne réponse était « Voila un texte avec plein plein de  
doublons, mais une petite macro bien faite devrait les  
supprimer. ». En effet, ce code n'utilisant pas trim, il compare les mots deux à  
deux, mais en prenant en compte les espaces pour la comparaison, ce qui fait que le  
doublon plein n'est pas reconnu comme tel puisque sa deuxième occurrence est suivie  
de 3 espaces alors que la première n'est suivie que d'un espace.

## 6. Divers

a. Indentation : pourquoi faut-il indenter son code ?

(a) Car sinon Visual Basic ne voudra pas compiler (b) Pour faciliter la relecture du code

(c) Car sinon Visual Basic risque de mal comprendre certaines instructions (d) Car sinon Visual Basic va mal  
comprendre certaines instructions

La bonne réponse était « pour faciliter la relecture du code ». L'indentation ne modifie  
en rien l'interprétation qui est faite par Visual Basic du Code mais sert aux  
programmeurs à relire leur code le plus facilement possible.

- b. A quoi sert l'option `explicit` au début des modules Visual Basic ?
- (a) A rendre toutes les variables globales
  - (b) Force à déclarer toutes les variables utilisées
  - (c) Permet d'utiliser des variable sans les déclarer
  - (d) Permet d'activer l'auto-complétion

La bonne réponse était « force à déclarer toutes les variables utilisées ». Ceci est très utile pour éviter de se tromper en écrivant le nom d'une variable, car, le cas échéant, Visual Basic vous avertira de l'utilisation illicite d'une variable non déclarée.