Chapitre 4 : Procédures et fonctions, Tableaux

Emmanuel Hyon
issus de L. Mesnager

Université de Paris Ouest Nanterre la Défense

2014-2015

Sommaire

Sommaire

Qu'est qu'une procédure ?

Il arrive parfois qu'on soit amené à répéter du code à plusieurs endroits d'un programme :

```
Sub Programme()
s = UCase(InputBox("Etes-vous marié ?"))
While s <> "OUI" AND s <> "NON"
    MsgBox "Répondez oui ou non SVP"
    s = UCase(InputBox("Etes-vous marié ?"))
Wend
If s = "OUI" Then MsgBox "Félicitations"

s = UCase(InputBox("Avez-vous des enfants ?"))
While s <> "OUI" AND s <> "NON"
    MsgBox "Répondez oui ou non SVP"
    s = UCase(InputBox("Avez-vous des enfants ?"))
Wend
If s = "OUI" Then MsgBox "Félicitations"
End Sub
```

Qu'est qu'une procédure ?

Dans le programme précédent, on répète en fait deux la même séquence d'instructions qui consiste à poser une question dont la réponse est oui ou non (en répétant la question si l'utilisateur donne une autre réponse que oui ou non), Une autre manière d'écrire le programme serait d'introduire une variable auxiliaire `Question`

```
Dim Question As String

Question="Etes-vous marié ?"

s = UCase(InputBox(Question))
While s <> "OUI" AND s <> "NON"
    MsgBox "Répondez oui ou non SVP"
    s = UCase(InputBox(Question))
Wend
If s = "OUI" Then MsgBox "Félicitations"
```

Qu'est qu'une procédure ?

En d'autres termes, on peut remplacer le programme précédent par

```
1  Dim Question As String
2  Question="Etes-vous marié ?"
3  s = UCase(InputBox(Question))
4  While s <> "OUI" AND s <> "NON"
5      MsgBox "Répondez oui ou non SVP"
6      s = UCase(InputBox(Question))
7  Wend
8  If s = "OUI" Then MsgBox "Félicitations"
9  Question="Avez-vous des enfants ?"
10 s = UCase(InputBox(Question))
11 While s <> "OUI" AND s <> "NON"
12     MsgBox "Répondez oui ou non SVP"
13     s = UCase(InputBox(Question))
14 Wend
15 If s = "OUI" Then MsgBox "Félicitations"
```

Qu'est qu'une procédure ?

Dans le programme précédent, on peut alors remarquer que le programme contient deux fois la même séquence d'instruction (les lignes 2 à 8 sont répétées une deuxième fois entre les lignes 10 et 14).

Dans ce type ce type de situation, on préférera alors créer ce qu'on appelle une **procédure** (un sous-programme) contenant cette séquence d'instruction, c'est-à-dire donner un nom à cette séquence d'instruction.

De plus, on voit que cette séquence d'instruction dépend de la valeur d'une variable `Question` de type `String`.

Qu'est qu'une procédure ?

Pour évider de répéter deux fois la même séquence d'instruction dans le programme, on va alors déclarer une procédure ReponseOuiNon contenant la séquence d'instructions.

Pour déclarer une procédure, on utilise le mot-clé **Sub**.

Pour indiquer que le comportement de la procédure dépend d'une variable `Question` de type `String`, on écrit en VB alors

```
Sub ReponseOuiNon(Question As String)
```

Vocabulaire. On dira alors que la procédure a un argument nommé `Question` de type `String`.

Qu'est qu'une procédure ?

La définition complète de la procédure est alors :

```
Sub ReponseOuiNon(Question As String)
    Dim s As String
    s = UCase(InputBox(Question))
    While s <> "OUI" AND s <> "NON"
          MsgBox "Répondez␣oui␣ou␣non␣SVP"
          s = UCase(InputBox(Question))
    Wend
    If s = "OUI" Then MsgBox "Félicitations"
End Sub
```

Qu'est qu'une procédure ?

Une écriture équivalente du programme précédent est alors la suivante :

```
Call ReponseOuiNon("Etes-vous marié ?")
Call ReponseOuiNon("Avez-vous des enfants ?")
```

Qu'est qu'une procédure ?

En résumé, une procédure est un sous-programme qui dépend d'une ou de plusieurs variables : les **arguments** de la procédure.

Une procédure peut avoir un nombre quelconque d'arguments. Dans ce cas, on sépare la déclaration des arguments par une virgule

```
Sub f(x As Integer, y As Integer)
```

Pour utiliser une procédure dans un programme, on utilise le mot-clé **Call** en donnant le nom de la procédure et les valeurs des arguments.

```
Call f(3,5)' x = 3 et y = 5
Call f(5,3) 'x = 5 et y = 3
```

Important. L'ordre dans lequel on donne les valeurs en argument de la procéure est important !

Qu'est qu'une procédure ?

On peut aussi donner en argument d'une prcoédure le nom d'une variable

```
Dim u As Integer, v As Integer

Call f(u,3) 'x=u et y=3
Call f(1,v) 'x=1 et y=v
Call f(u,v)
```

Qu'est qu'une procédure ?

Dans la définition d'une procédure, on a le droit d'utiliser les arguments dans des calculs intermédiaires.

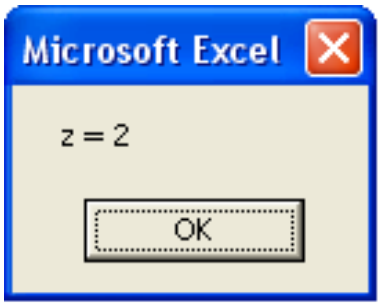
```
Sub f(x As Integer, y As Integer)
y = 2 * x
MsgBox y
End Sub
```

Qu'est qu'une procédure ?

Mais dans ce cas, si on écrit

```
Dim x As Integer, z As Integer
x = 1
z = 5
Call f(x, z)
MsgBox z
```

Alors la valeur de la variable z sera 2 suite à l'appel de la procédure f !



Qu'est qu'une procédure ?

Pourquoi ?

En fait, en passant la variable `z` en argument de la procédure, VB considère que l'argument `y` de `f` et la variable `z` sont deux noms différents pour le même espace mémoire.

On dit alors que l'argument `y` est passé par référence.

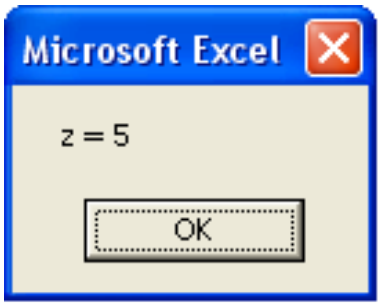
Qu'est qu'une procédure ?

Si l'on souhaite que la variable `z` et l'argument `x` soient considérées comme des variables séparées (ne partageant pas le même espace en mémoire), on doit utiliser lemot-clé `ByVal`

```
Sub f(x As Integer, ByVal y As Integer)
y = 2 * x
MsgBox y
End Sub
```

On dit alors que l'argument `y` est passé par valeur

Dans ce cas, la valeur de la variable `z` n'est pas modifée suite à l'appel de la procédure `f`



Qu'est qu'une procédure ?

En VB, on peut utiliser le mot-clé `ByRef` si l'on souhaite souligner le fait qu'un argument est passé par référence (mais ce n'est pas obligatoire)

```
Sub f(ByRef x As Integer, ByVal y As Integer)
```

- L'argument `x` est passé par référence
- L'argument `y` est passé par valeur.

Qu'est qu'une procédure ?

On a vu que lors de l'appel d'une procédure, l'ordre dans lequel on donne les arguments est importent.

En VB, on peut les donner dans n'importe quel ordre de la manière suivante en utilisant les noms des arguments :

```
Dim x As Integer, z As Integer
x = 1
z = 5
Call f(y:=z,x:=1)
MsgBox z
```

On indique chaque argument de la procédure en écrivant

`nomArgument := Valeur ou Variable`

Qu'est qu'une procédure ?

On peut aussi indiquer à l'aide du mot-clé **Optional** que certains arguments sont optionnels :

```
Sub g(x As Integer, Optional s As String)
    MsgBox s & "␣" & x
End Sub
```

On peut alors appeler la procédure g en écrivant

```
Call g(1,"Master") 'affiche Master 1 dans une boîte de dialog
```

Qu'est qu'une procédure ?

On peut aussi appeler la procédure g sans l'argument optionnel. Par exemple, l'instruction

```
Call g(2)
```

affiche " 2" dans une boîte de dialogue.
Remarque. Par défaut, une chaîne de caractères est initialisée à "" (la chaîne vide)

Qu'est qu'une procédure ?

On peut aussi donner une valeur par défaut un argument optionnel

```
Sub g(x As Integer, Optional s As String = "Licence")
    MsgBox s & "␣" & x
End Sub
```

Dans ce cas, l'instruction

```
Call g(2)
```

affiche `"Licence 2"` dans une boîte de dialogue.

Explication. La valeur de l'argument `s` n'étant pas précisé lors de l'appel de la procédure g, l'argument `s` prend la `"licence"` (valeur par défaut).

Sommaire

Qu'est-ce qu'une fonction ?

Une **fonction** est une « procédure » retournant un résultat. Une fonction est déclarée à l'aide du mot-clé **Function**

```
Function SurfaceCercle(r As Double) As Double
    Const PI As Double = 3.14
    SurfaceCercle = PI * r ^ 2
End Function
```

Alors si on écrit

```
s = SurfaceCercle(4)
```

La variable s recevra la valeur 3.14×4^2

Qu'est-ce qu'une fonction ?

Une fonction peut avoir un nombre quelconque d'arguments ou ne pas avoir d'arguments

```
Function g(x As Integer, s As String) As String
Function h() As Boolean
```

Important. On doit préciser le type de valeurs retournée par la fonction.

Qu'est-ce qu'une fonction ?

Pour indiquer la valeur retournée par la fonction, on écrit

```
NomDeLaFonction = Valeur
```

Important. Toutes les instructions qui suivent sont exécutées !

```
Function h()
    h = 2.0
    MsgBox "Bonjour" 'l'instruction est exécutée
End Function
```

Sommaire

Tableaux de taille fixe

Un tableau est une variable dans laquelle on peut stocker plusieurs valeurs du même type. De plus, on peut accéder à chacune des valeurs à l'aide d'un numéro qu'on appelle son indice.

Pour déclarer un tableau, on écrit

```
Dim NomTableau(Debut To Fin) As Type
```

Exemple.

```
Dim t(1 To 7) As Integer 'tableau de 7 entiers
'numérotés de 1 à 7

Dim p(2 To 4) As String
'tableau de 3 chaînes de caractères
numérotées de 2 à 4
```

Tableaux de taille fixe

Pour accéder à une des valeurs contenues dans le tableau, on écrit

```
NomTableau(indice)
```

Exemple.

```
t(5) = 7

p(3) = "Master"
```

Tableaux de taille fixe

Il existe deux fonctions `LBound` et `UBound` qui renvoient respectivement le plus petit indice et le plus grand indice

```
Dim t(1 To 5) As Integer

Dim p(2 To 6) As String

LBound(t) 'renvoie 1

UBound(t) 'renvoie 5

LBound(p) 'renvoie 2

UBound(p) 'renvoie 6
```

Tableaux à plusieurs dimensions

On peut définir des tableaux à plusieurs dimensions

```
Dim t(1 To 7,2 To 9) As Long
```

Pour accéder à une des cases du tableau, on écrit alors

```
t(i,j)

'i doit être compris entre 1 et 7
'j doit être compris entre 2 et 9
```

Tableaux à plusieurs dimensions

Pour connaître le plus indice et le plus grand indice, on peut utiliser les fonctions `LBound` et `UBound` avec un argument supplémentaire : 1 pour le première dimension, 2 pour la seconde dimension, etc ...

```
MsgBox LBound(t,1) & "␣To␣" & UBound(t,1)
'affiche "1 To 7"

MsgBox LBound(t,2) & "␣To␣" & UBound(t,2)
'affiche "2 To 9"
```

Tableaux de taille variable

On peut déclarer des tableaux de taille variable

```
Dim t() As Long
```

On peut alors dimensionner (ou redimensionner) le tableau à l'aide de l'instruction `ReDim`

```
ReDim t(1 To 3) As Long
```

Remarque. On peut redimensionner plusieurs fois le même tableau.

Tableaux de taille variable

Attention. Lors d'un redimensionnement, le contenu du tableau est réinitialisé à 0.

Pour redimensionner un tableau de taille variable sans effacer son contenu, on rajoute le mot-clé `Preserve`.

```
Dim t() As Integer

ReDim t(2 To 9) As Integer

t(2) = 4

ReDim Preserve t(2 To 14) As Integer
```

Tableaux en arguments de procédures ou fonctions

On peut définir une procédure ou d'une fonction prenant en argument un tableau.

```
Sub f(t() As Integer)
```

Attention. On ne précise pas la dimension du tableau.

Sommaire

Stocker dans un tableau les valeurs contenues dans une plage de cellules

On peut directement stocker dans un tableau de `Variant` toutes les valeurs d'une plage de cellules.

```
Dim v() As Variant

v = [A1:B5].Value
v = Range("A1:B5").Value
```

Après affectation, `v` est un tableau à deux dimensions : la première dimension est égale au nombre de lignes de la plage et la second au nombre de colonnes de la plages.

Ecrire le contenu d'un tableau dans une plage de cellules

On peut aussi directement affecter à toutes les cellules d'une plage des valeurs contenues dans un tableau.

```
Dim t(1 To 3,1 To 2) As Variant

t(1,1) = True
t(1,2) = 2.14
t(2,1) = 2
t(2,2) = CDate("11/10/2010")
t(3,1) = "chaîne"
t(3,2) = 50000

[E1:F3].Value = t
Range("E1:F3").Value = t
```

Attention. Le nombre de lignes de la plage et la première dimension du tableau doivent être égales. De même, le nombre de colonnes du tableau et la deuxième dimension du tableau doivent être égales.

Copier une plage de cellules

Pour copier les valeurs d'une plage de cellules dans une autre plage de cellules de même dimension, une man ière de faire est d'écrire :

```
[H7:I9].Value = [E1:F3].Value
Range("H7:I9").Value = [E1:F3].Value
Range("H7:I9").Value = Range("E1:F3").Value
```

Attention. Cette méthode ne copie que les valeurs pas les formules !

Copier une plage de cellules

Une autre manière consiste à utiliser la méthode `Copy` :

```
[E1:F3].Copy Destination:=[H7:I9]
```

Le contenu (valeur et formule) de chaque cellule de la plage `E1:F3` est copié dans les cellules de la plage `H7:I9`.

Attention. La méthode actualise les références relatives contenues dans les formules de la plage copiée lors de la copie.