Chapitre 1 : Programmer en Visual Basic, les premiers pas.

Emmanuel Hyon

Université de Paris Ouest Nanterre la Défense

2013-2014

Sommaire

Sommaire

Définition et vocabulaire

Définition.

Une **variable** est un emplacement mémoire dans lequel on stocke des données d'un certain type (valeurs numériques entières, décimales, chaînes de caractères, etc ...).

Définition et vocabulaire

Définition.

Une **variable** est un emplacement mémoire dans lequel on stocke des données d'un certain type (valeurs numériques entières, décimales, chaînes de caractères, etc ...).

Lorsqu'on définit une variable, il est très important de :

- définir le nom de la variable, c'est-à-dire, son **identifiant**,

Définition et vocabulaire

Définition.

Une **variable** est un emplacement mémoire dans lequel on stocke des données d'un certain type (valeurs numériques entières, décimales, chaînes de caractères, etc ...).

Lorsqu'on définit une variable, il est très important de :

- définir le nom de la variable, c'est-à-dire, son **identifiant**,
- préciser le type de valeur que prend la variable : le **type de données**.

Identifiant d'une variable

Pour définir l'identifiant d'une variable, on doit respecter les règles suivantes :

- commencer par un caractère alphabétique,
- utiliser uniquement des caractères alphabétiques, des chiffres décimaux et le caractère _

Identifiant d'une variable

Pour définir l'identifiant d'une variable, on doit respecter les règles suivantes :

- commencer par un caractère alphabétique,
- utiliser uniquement des caractères alphabétiques, des chiffres décimaux et le caractère _

Exemples d'identifiants valides.

```
x
valeur
a_1
a1
valeur2011
```

Identifiants

Attention. La casse n'est pas pris en compte !
Les identifiants ci-dessous seront considérés comme étant identiques en `Visual Basic` :

`nom` `Nom` `NOM` `NoM`

Un conseil : n'utilisez que des lettres minuscules.

Identifiants

Remarque. Lors de la définition d'une variable, la validité de l'identifiant utilisé est vérifié.

Les types de données

Il existe de nombreux types de données en `Visual Basic` :

- les nombres entiers : `Byte`, `Integer`, `Long`, ...

  `2011` `0` `-1` `67`

Les types de données

Il existe de nombreux types de données en `Visual Basic` :

- les nombres entiers : `Byte`, `Integer`, `Long`, ...

  `2011` `0` `-1` `67`

- les nombres décimaux : `Single`, `Double`, `Currency`, ...

  `3.14` `1E+17` `-1.6` `1.45E-6` `2011#` `0.0`

Les types de données

Il existe de nombreux types de données en `Visual Basic` :

- les nombres entiers : `Byte`, `Integer`, `Long`, ...

  `2011` `0` `-1` `67`

- les nombres décimaux : `Single`, `Double`, `Currency`, ...

  `3.14` `1E+17` `-1.6` `1.45E-6` `2011#` `0.0`

- les chaînes de caractères (délimitées par le caractère `"`) : `String`.

  `"Master ISIFAR"` `"M"` `"Laurent.Mesnager@u-paris10.fr"`

Les types de données

Il existe de nombreux types de données en `Visual Basic` :

- les nombres entiers : `Byte`, `Integer`, `Long`, ...

  `2011` `0` `-1` `67`

- les nombres décimaux : `Single`, `Double`, `Currency`, ...

  `3.14` `1E+17` `-1.6` `1.45E-6` `2011#` `0.0`

- les chaînes de caractères (délimitées par le caractère `"`) : `String`.

  `"Master ISIFAR"` `"M"` `"Laurent.Mesnager@u-paris10.fr"`

- les valeurs logiques : `Boolean`.

  `True` `False`

Les types de données

Il existe de nombreux types de données en `Visual Basic` :

- les nombres entiers : `Byte`, `Integer`, `Long`, ...

  `2011` `0` `-1` `67`

- les nombres décimaux : `Single`, `Double`, `Currency`, ...

  `3.14` `1E+17` `-1.6` `1.45E-6` `2011#` `0.0`

- les chaînes de caractères (délimitées par le caractère `"`) : `String`.

  `"Master ISIFAR"` `"M"` `"Laurent.Mesnager@u-paris10.fr"`

- les valeurs logiques : `Boolean`.

  `True` `False`

- etc ...

L'instruction **Dim**

Pour déclarer une variable en `Visual Basic`, on utilise le mot-clé **Dim**. la forme générale d'une déclaration de variables est :

```
Dim identifiant_de_la_variable As Type_de_donnes
```

L'instruction Dim

Pour déclarer une variable en `Visual Basic`, on utilise le mot-clé **Dim**. la forme générale d'une déclaration de variables est :

```
Dim identifiant_de_la_variable As Type_de_donnes
```

Exemple.

```
Dim i As Integer
Dim x As Double
Dim s As String
Dim b As Boolean
```

L'instruction **Dim**

Remarque. On peut déclarer plusieurs variables d'un coup en séparant les déclarations par une virgule.

```
Dim i As Integer, x As Single, c As currency
```

L'affectation

Pour donner une valeur à une variable, on utilise l'opérateur = (appelé **opérateur d'affectation**).

Exemple.

```
Dim i As Integer, x As Double, s As String
i = 2010
x = 3.25
s = "Master ISIFAR"

Dim b As Boolean, c As Currency
b = True
c = 1.25
```

L'affectation

Pour lire la valeur d'une variable, il suffit d'utiliser son identifiant :

Exemple.

```
Dim i As Integer, j As Integer

i = 2010
j = i
```

Les variables `i` et `j` valent toutes les deux 2010.

Attention avec les variables numériques !

Attention. Pour chaque type de données entier, l'intervalle des valeurs possibles varie !

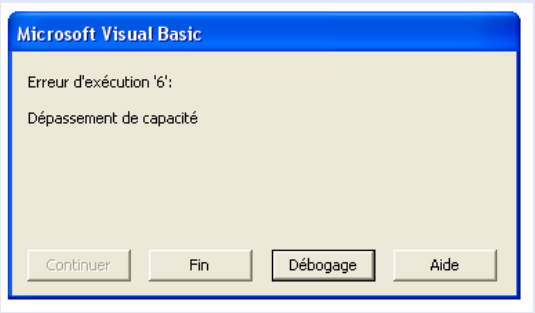
- un `Byte` varie entre 0 et 255,
- un `Integer` entre -32768 et 32767,
- un `Long` entre -2147483648 et 2147483647

Attention avec les variables numériques !

Attention. Pour chaque type de données entier, l'intervalle des valeurs possibles varie !

- un `Byte` varie entre 0 et 255,
- un `Integer` entre -32768 et 32767,
- un `Long` entre -2147483648 et 2147483647

Remarque. Si vous essayez de donner une valeur trop grande ou trop négative à une variable entière (par exemple 500 pour un `Byte`), vous risquez de voir surgir la fenêtre de message :



Attention avec les variables numériques !

Remarque. De même, une variable décimale ne peut pas prendre n'importe quelle valeur.
Comme pour les variables entières, attention aux dépassements de capacité !

Opérations arithmétiques sur les types numériques

Le langage `Visual Basic` fournit plusieurs opérateurs arithmétiques binaires :

Opérateur	**Opération**
+	Addition
*	Multiplication
-	Soustraction
/	Division décimale
^	Élévation à la puissance (`10^3` renvoie 1000)
\	Retourne le quotient de la division euclidienne (l'opération `20 \ 3` renvoie 6)
`Mod`	Retourne le reste de la division euclidienne (l'opération `20 Mod 3` retourne 2).

Expressions arithmériques

Il est possible de former des expressions arithmétiques

```
Dim i As Integer, j As Integer

j = 4
i =  4 * j + 5
```

Expressions arithmériques

Il est possible de former des expressions arithmétiques

```
Dim i As Integer, j As Integer

j = 4
i =  4 * j + 5
```

Remarque. Il existe des règles de priorité entre les différents opérateurs. Par exemple, l'opérateur de multiplication est prioritaire devant l'opérateur d'addition.

On peut utiliser les parenthèses ouvrantes et fermantes pour grouper une expression :

```
Dim k As Integer
k = 4 * (i + 5)
```

Concaténation des chaînes de caractères

Définition.

La concaténation de plusieurs chaînes de caractères est l'opération qui consiste à mettre bout à bout les chaînes de caractères. En `Visual Basic`, l'opérateur de concaténation est le caractère `&`.

Concaténation des chaînes de caractères

Définition.

La concaténation de plusieurs chaînes de caractères est l'opération qui consiste à mettre bout à bout les chaînes de caractères. En `Visual Basic`, l'opérateur de concaténation est le caractère `&`.

Exemple.

```
Dim s As String
s = "Master ISIFAR" & " " & "2012-2013"
```

La variable `s` contient la chaîne de caractères `"Master ISIFAR 2012-2013"`.

Conversions automatiques

Si vous donnez à une variable une valeur d'un autre type que celui de la variable, la valeur sera automatiquement convertie en une valeur du type de la variable.

Conversions automatiques

Si vous donnez à une variable une valeur d'un autre type que celui de la variable, la valeur sera automatiquement convertie en une valeur du type de la variable.

Exemple.

```
Dim s As String

s = 1 's contient la chaîne de caractères "1"

s = True 's contient la chaîne de caractères "Vrai"

s = 3.14 's contient la chaîne de caractères "3,14"
```

Conversions automatiques

Exemple.

```
Dim i As Integer, b As Boolean, x As Double

i = "1" 'i vaut 1

b = "Vrai" 'b vaut True

x = "3,14" 'x vaut 3.14
```

Fonctions de conversions

La langage `Visual Basic` fournit un certain nombre de fonctions pour convertir une valeur d'un certain type en un autre type.

Exemple.

`Str(nombre)`	convertit un nombre en une chaîne de caractères
`CInt(s)`	convertit une chaîne de caractères en un `Integer`
`CDbl(s)`	convertit une chaîne de caractères en un `Double`
`CBool(s)`	convertit une chaîne de caractères en un `Boolean`

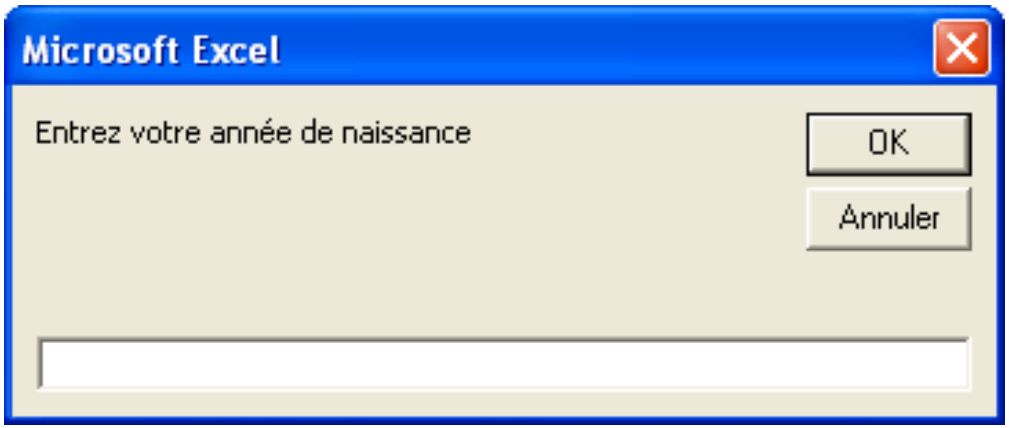
Sommaire

La commande InputBox

La fonction InputBox permet d'afficher une boite de dialogue contenant une zone de texte dans laquelle on peut saisir une valeur

```
Dim annee As Integer

annee = InputBox ("Entrez votre date de naissance")
```



Attention : La valeur retournée par la fonction InputBox est une chaîne de caractères. Donc, dans l'affectation ci-dessus, la chaîne de caractères est convertie en un nombre entier (si cela est possible !).

La commande MsgBox

La commande MsgBox permet d'afficher un message dans une fenêtre.

Msgbox "Ceci␣est␣un␣message"

Sommaire

Définir une macro

Définition.

Une **macro** est un terme générique pour désigner un moyen de mémoriser un enchaînement de traitements au sein d'un logiciel.
Une fois mémorisé, Il est ensuite possible d'utiliser la macro –l'expression standard est « lancer la macro » –pour répéter le même enchaînement.

Définir une macro

Définition.

Une **macro** est un terme générique pour désigner un moyen de mémoriser un enchaînement de traitements au sein d'un logiciel.
Une fois mémorisé, Il est ensuite possible d'utiliser la macro –l'expression standard est « lancer la macro » –pour répéter le même enchaînement.

Pour définir et délimiter une macro `Visual Basic`, on utilise les mot-clés **Sub** et **End**.

Définir une macro

Définition.

Une **macro** est un terme générique pour désigner un moyen de mémoriser un enchaînement de traitements au sein d'un logiciel.
Une fois mémorisé, Il est ensuite possible d'utiliser la macro –l'expression standard est « lancer la macro » –pour répéter le même enchaînement.

Pour définir et délimiter une macro `Visual Basic`, on utilise les mot-clés **Sub** et **End**.

```
Sub nom_macro()
    'déclarations des variables

    'instructions
End Sub
```

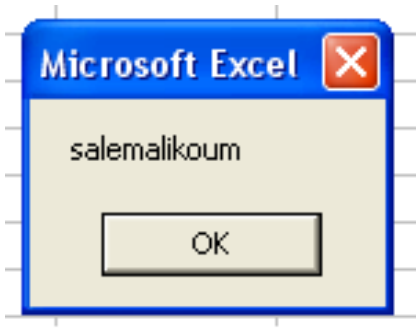
Une première macro

```
Sub Salemalikoum()

MsgBox "salemalikoum"

End Sub
```

affiche le message suivant dans une fenêtre



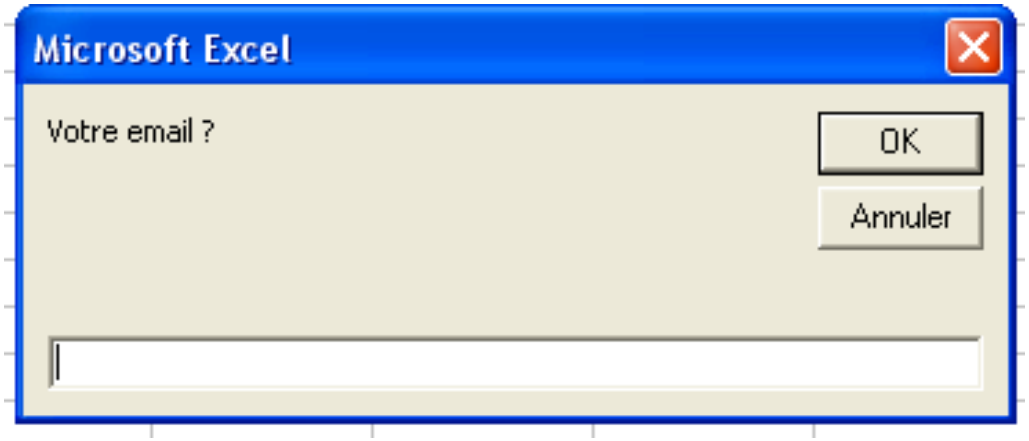
Une macro avec variables

```
Sub monemail()
Dim email As String

email = InputBox("Votre email ?")

MsgBox "Ecrivez-moi à " & email
End Sub
```

La macro commence par afficher la boîte de dialogue



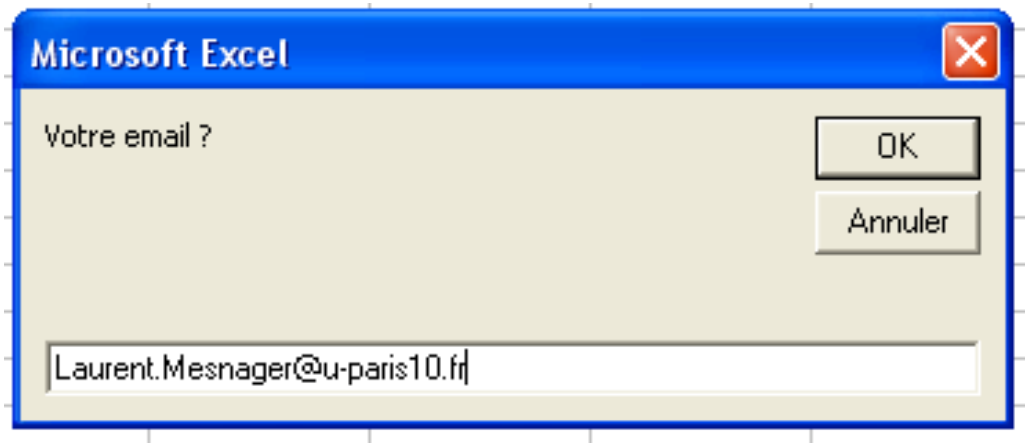
Une macro avec variables

```
Sub monemail()
Dim email As String

email = InputBox("Votre email ?")

MsgBox "Ecrivez-moi à " & email
End Sub
```

L'utilisateur saisit son courrier électronique



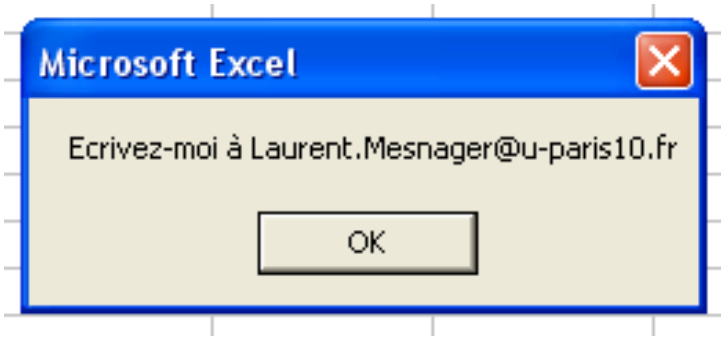
Une macro avec variables

```
Sub monemail()
Dim email As String

email = InputBox("Votre email ?")

MsgBox "Ecrivez-moi à " & email
End Sub
```

La macro affiche le message suivant dans une fenêtre



Une autre macro avec variables

```
Sub note()

Dim noteTD As Double, noteQCM As Double
Dim examen As Double,moyenne As Double

noteTD = 12 : noteQCM = 10 : examen = 8

moyenne = 0.3 * noteTD + 0.3 * noteQCM + 0.4 * examen

MsgBox "Votre moyenne est : " & moyenne

End Sub
```

Une autre macro avec variables

```
Sub note()

Dim noteTD As Double, noteQCM As Double
Dim examen As Double,moyenne As Double

noteTD = 12 : noteQCM = 10 : examen = 8

moyenne = 0.3 * noteTD + 0.3 * noteQCM + 0.4 * examen

MsgBox "Votre moyenne est : " & moyenne

End Sub
```

Remarque. Nous avons utilisé dans l'écriture de la macro le caractère : qui permet de mettre sur la même ligne plusieurs instructions.

Sommaire

L'instruction `If .. Then`

On peut conditionner l'exécution d'une série d'instructions au fait qu'une certaine condition est remplie.

```
If condition Then
    Série d'instructions
End If
```

où condition est un expression booléenne, c'est-à-dire une expression qui a pour valeur `True` ou `False`.

L'instruction If .. Then

Exemple.

```
Sub Inverse()

Dim x As Integer
x = InputBox("Entrez un nombre entier")

If x <> 0 Then 'x différent de 0
    MsgBox "L'inverse de " & x & " est " & 1/x
End If

End Sub
```

Ce programme demande à l'utilisateur un nombre entier puis affiche, s'il est non nul, son inverse dans une boîte de dialogue.

Opérateurs de comparaison

Pour comparer deux valeurs numériques, le langage VB fournit plusieurs opérateurs de comparaison : =, <, >, <=, >=, <>

Opérateur	Comparaison
=	égal à
<>	différent de

Exemple.

```
Dim x As Integer
x = 2009
```

L'évaluation des conditions `x = 2009` et `x <> 2009` renvoie, respectivement, `True` et `False`.

Opérateurs de comparaison

Pour comparer deux valeurs numériques, le langage VB fournit plusieurs opérateurs de comparaison : =, <, >, <=, >=, <>

Opérateur	Comparaison
<	strictement inférieur à
<=	inférieur ou égal à
>	strictement supérieur à
>=	supérieur ou égal à

Exemple.

```
Dim x As Integer
x = 3
```

L'évaluation des conditions `x < 3` et `x <= 3` renvoie, respectivement, `False` et `True`.

L'instruction If ... Then ... Else ...

Il est possible de ne pas exécuter les mêmes séries d'"instructions selon qu'une condition soit vraie ou fausse.

```
If condition Then
    Série d instructions 1
Else
    Série d instructions 2
End If
```

L'exécution du programme ci-dessus est le suivant :

- Si la condition est vraie alors la série d'instructions 1 est exécutée.

L'instruction If ... Then ... Else ...

Il est possible de ne pas exécuter les mêmes séries d'"instructions selon qu'une condition soit vraie ou fausse.

```
If condition Then
    Série d instructions 1
Else
    Série d instructions 2
End If
```

L'exécution du programme ci-dessus est le suivant :

- Si la condition est vraie alors la série d'instructions 1 est exécutée.
- Si la condition est fausse alors la série d'instructions 2 est executée.

L'instruction If ... Then ... Else ...

Exemple.

```
If Date < CDate("08/10/2010") Then
    MsgBox "Vous avez encore le temps !"
Else
    MsgBox "Attention ! plus que 7 jours "
End If
```

Remarque. La fonction Date renvoie la date courante.

L'instruction If ... Then ... Else ...

Si la date courante est antérieure au 8 octobre 2010, le programme affiche la boîte de dialogue

L'instruction If ... Then ... Else ...

Explication. La condition

```
Date < CDate}("08/10/2010")
```

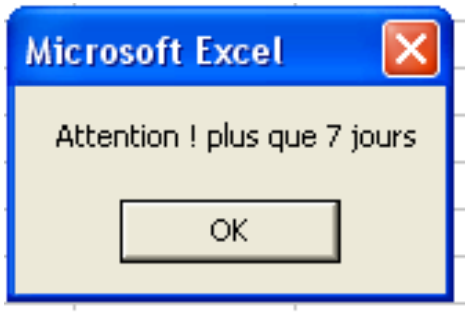
est vraie, c'est donc l'instruction

```
If Date < CDate("08/10/2010") Then
    MsgBox "Vous avez encore le temps ! "
Else
    MsgBox "Attention ! plus que 7 jours "
End If
```

qui est executée

L'instruction `If ... Then ... Else ...`

Si la date courante (renvoyée par la fonction **Date**) est postérieure au 8 octobre 2010, le programme affiche la boîte de dialogue



L'instruction If ... Then ... Else ...

Explication. La condition

```
Date < CDate("08/10/2010")
```

est fausse ; c'est donc l'instruction

```
If Date < CDate("08/10/2010") Then
    MsgBox "Vous avez encore le temps ! "
Else
    MsgBox "Attention ! plus que 7 jours "}
End If
```

qui est executée

Opérateurs booléens

On peut combiner plusieurs conditions à l'aide des opérateurs booléens : `And`, `Or` and `Not`

Opération	**Opérateur**	**Description**
Disjonction	`Or`	`a Or b` vaut `True` si au moins une des deux variables `a` et `b` vaut `True` et `False` sinon
Conjonction	`And`	`a And b` vaut `True` si les deux variables `a` et `b` valent `True` et `False` sinon
Négation	`Not`	`Not a` vaut `True` si `a` vaut `False` et `False` sinon

Opérateurs booléens

Exemple.

```
Dim moyenne_M2 As Single, note_stage As Single
Dim recu As Boolean

If moyenne_M2 >= 10 And note_stage >= 10 Then
    recu = True
Else
    recu = False
End If
```

Un étudiant n'est recu en Master 2 que s'il a une moyenne générale supérieure ou égale à 10 et une note supérieure ou égal à 10 au stage sinon il est ajourné.

Conditions imbriquées

On peut imbriquer plusieurs instructions `If ... Then`

Exemple.

```
If age < 18 Then
    MsgBox "ne peut voter"
Else
    If inscrit = False Then
        MsgBox "ne peut voter"
    Else
        MsgBox "peut voter"
    End If
End If
```

Conditions imbriquées

Le langage `Visual Basic` autorise un nombre indéterminé de conditions dans une structure `If ... Then ... Else`. On utilise pour cela le mot-clé **ElseIf**.

Exemple.

```
If age < 18 Then
    MsgBox "ne peut voter"
ElseIf inscrit = False Then
    MsgBox "ne peut voter"
Else
    MsgBox "peut voter"
End If
```

Sommaire

L'instruction Select

On peut sélectionner selon la valeur d'une expression différentes séries d'instructions à exécuter

```
Select Case expression

Case ensemble_valeurs_1
    Serie d'instructions 1

Case ensemble_valeurs_2
    Serie d'instructions 2

...

Case ensemble_valeurs_N
    Serie d'instructions N
End Select
```

L'instruction Select

Exemple.

```
Dim i As Integer, j As Integer

...

Select Case 2 * i + 1
Case 1
    j = 2
Case 2, 3, 4
    j = 3
Case 5 To 67
    j = 4
Case Is >= 68
    j = 5
End Select
```

L'instruction Select

Exemple.

```
Dim nationalite As String

...

Select Case nationalite

Case Is = "francais"
    MsgBox "Bonjour"
Case Is = "tunisien"
    MsgBox "Aslema"

End Select
```

L'instruction Select

Il existe une clause optionnelle **Case Else** qui permet de spécifier une série d'instructions à exécuter si aucun des **Case**s n'a été executé :

```
Dim nationalite As String

...

Select Case nationalite

Case Is = "francais"
    MsgBox "Bonjour"
Case Is = "tunisien"
    MsgBox "Aslema"

Case Else
    MsgBox "Hello"
End Select
```

Sommaire

L'instruction For

L'instruction **For** permet de répéter un nombre fixé de fois une série d'instructions

```
Dim c As Integer

For c = valeur1 To valeur2
    Série d'instructions
Next c
```

La série d'instructions est exécutée en boucle en incrémentant la valeur du compteur c de 1 à chaque passage de la boucle.

L'instruction For

Exemple.

Le programme

```
Dim i As Integer
For i = 1 To 3
    MsgBox "Etape" & i
Next i
```

affiche les trois boîtes de dialogue suivantes l'une après l'autre.

L'instruction For

On peut incrémenter le compteur utilisé dans une boucle d'une autre valeur que 1 à l'aide du mot-clé Step

```
Dim i As Integer For i = 1 To 5 Step 2
    MsgBox "Etape " & i
Next i
```

Dans la boucle, la variable *i* prend successivement les valeurs 1, 3 et 5.

L'instruction For

On peut donner une valeur négative à l'incrément

```
Dim i As Integer
For i = 3 To 1 Step -1
    MsgBox "Etape " & i
Next i
```

La variable *i* prend successivement les valeurs 3, 2 et 1.

Imbrication de boucles For

On peut imbriquer deux boucles **For** :

```
For i = 1 To 4
    For j = 3 To 9

    ...

    Next j
Next i
```

Indices de boucle

Les indices de boucle peuvent être des expressions :

```
Dim j As Integer, i As Integer

...

For i = j To 2 * j

    ...

Next i
```

Sommaire

L'instruction **While**

On peut répéter une série d'instructions tant qu'une condition est vérifiée à l'aide de l'instruction **While**

```
While condition
    Série d'instructions
Wend
```

L'instruction While

Exemple.

```
Dim r As String
r = "a"
While r <> "."
    r = InputBox ("Tapez un caractère")
Wend
```

Tant que l'utilisateur entre un autre caractère que le point, on lui redemande sans cesse d'entrez un caractère dans la boîte de dialogue `InputBox`. Par contre, s'il entre le caractère `"."`, le programme s'arrête.