

Programmation VBA



Pierre BONNET

La programmation VBA

↪ Historiquement, la programmation sous Excel avait comme fonction d'automatiser une **succession d'actions** faites dans la feuille à l'aide de la souris.

D'où la notion de **Macro Excel**

Le langage initial était spécifique (versions en Anglais et en Français)

↪ A partir d'Excel 97, abandon du langage spécifique pour un langage unique adoptant la **syntaxe du Basic** :

Visual **B**asic for **A**pplication VBA

C'est un **enrichissement** de **VB** par des fonctions spécifiques à chaque application (Excel, Word, Access....).

↪ Excel comprend tous les outils d'écriture et d'exécution de VB, y compris les possibilités d'extension avec des bibliothèques ou "contrôles" supplémentaires.

↪ La différence essentielle est que l'affichage se fait dans les feuilles d'un classeur

Accès à la programmation VBA → Alt F11

The screenshot displays the Microsoft Excel interface with the Microsoft Visual Basic Editor (VBE) open. The Excel window shows a spreadsheet with the value 155 in cell A1. The VBE window shows the VBAProject for 'Tache_Cyclique.xls' with the following code:

```
Dim Heure_evenement As Date

Sub Tache_Cyclique()

    'Placer ici le code à exécuter avant le départ du dé

    Heure_evenement = Now + TimeSerial(0, 0, 1)
    Application.OnTime Heure_evenement, "Tache_Cyclique"

    'Placer ici le code à exécuter après le départ du dé

    Cells(1, 1) = Cells(1, 1) + 1

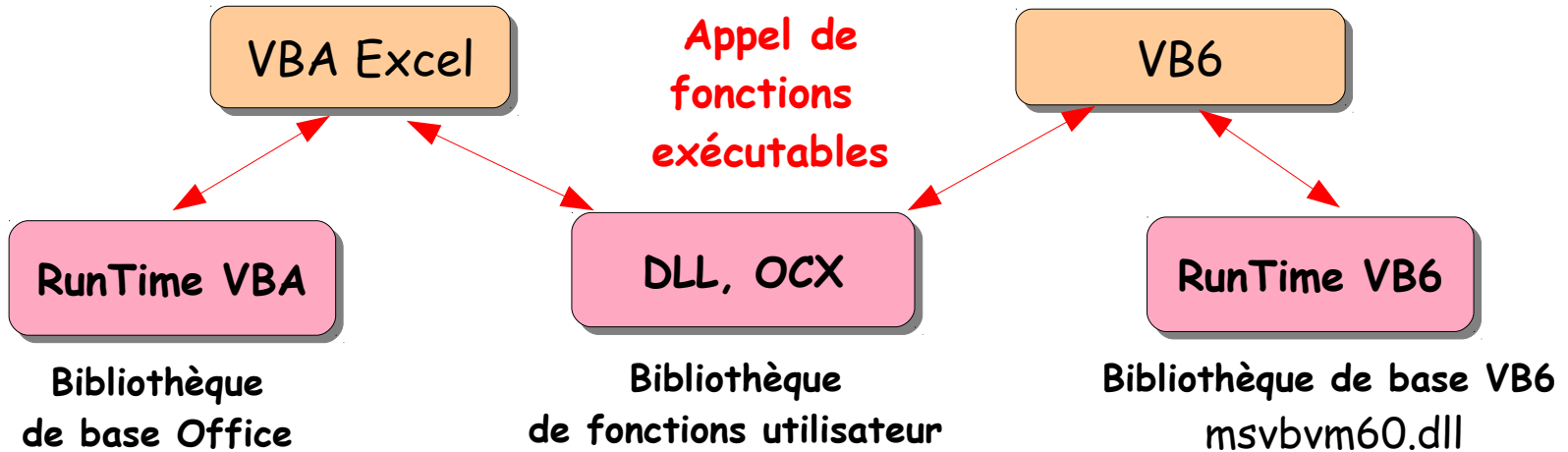
End Sub

Sub Arrêt_Tache_Cyclique()

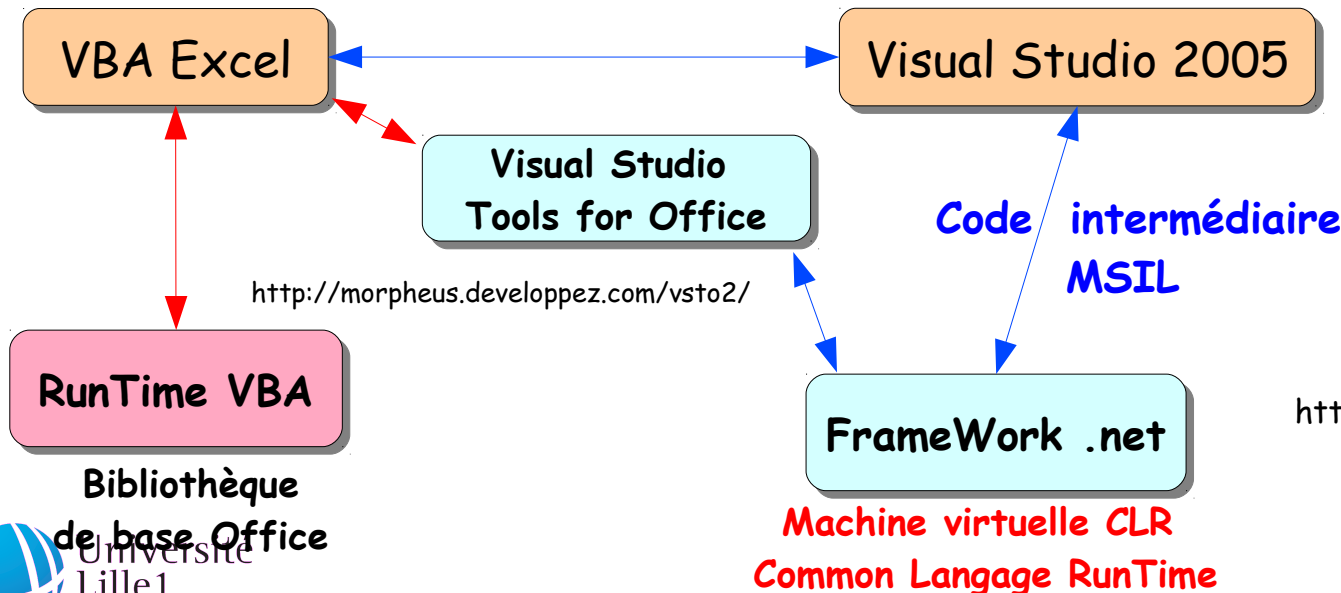
    On Error Resume Next
    Application.OnTime Heure_evenement, "Tache_Cyclique"

End Sub
```

VBA/VB6



FrameWork .NET



Environnement de développement
VB.NET, C#, C+
+.NET, Java,
J#, Python...



<http://fr.wikipedia.org/wiki/.NET>

Variables

● Déclaration implicite

Par défaut, l'usage d'une variable dans une affectation tient lieu de déclaration (comme en Matlab)

Le type affecté est déterminé automatiquement par l'évaluation de la partie droite de l'évaluation

Danger de l'absence de déclaration : toute erreur de frappe du nom d'une variable crée une nouvelle variable!

● Déclaration explicite

Utilisez la directive en début de programme

```
Option Explicit
```

Toute déclaration a la structure :

```
Dim Nom_Variable as Type_Variable
```

● Portée de la déclaration

Une variable n'est visible que pour la feuille où elle est déclarée .

Pour une visibilité des autres feuilles/classeur, utiliser `Public Nom_Variable as Type_Variable`

Particularité : La notion de pointeur n'existe pas en VB/VBA d'où communication impossible avec des fonctions qui utilisent cette approche pour le dialogue

Variables

- **Boolean**

Prend les valeurs True et False. Par conversion depuis d'autres types numériques, la valeur 0 devient False et toutes les autres deviennent True. Dans le cas inverse, False devient 0 et True devient -1.

Conversion d'une donnée numérique en Boolean par la fonction Cbool

exemple : A = CBool(10) 'A prend la valeur True.

- **Byte, Integer, Long**

Types de données entiers 8, 16 ou 32 bits .

Conversion d' une donnée numérique en Byte, utilisez la fonction CByte, idem pour les autres types .

Attention : l'adressage des lignes xls doit se faire en type long (valeur éventuellement supérieure à 32 767)

- **Single, Double**

Caractérise les nombres réels (ou flottants) simple et double précision

Variables

- **String** * Les chaînes de longueur fixe peuvent contenir de 1 à 2^{16} caractères.

```
Dim ma_chaine as string
ma_chaine = "Bonjour"
MsgBox ma_chaine
```

Ou bien `Dim ma_chaine$`

Il existe en VBA toutes les fonctions de conversion nombre <--> chaîne

voir `str` et `val`

concaténation de chaîne : `ma_chaine = "Bon" + "jour"`

`ma_chaine = "Amplitude :" & val(signal)`

- **Tableaux** Tous les types peuvent être étendus en tableau

```
Dim Tableau1(10) as double  
Tableau(3) = 3.14159
```

```
Dim MatriceXY(5,7) as Boolean      'dimension 60 au maximum  
MatriceXY(2,3) = True
```

L'indice de début est implicitement 0 par défaut , 1 en utilisant la déclaration `Option Base 1` ou en spécifiant les indices :

```
Dim MonTableau(1 To 5, 10 To 20) As String
```

Recherche des indices min et max par les fonctions `LBound` et `Ubound`

```
NbElements = (Ubound(MatriceXY,1)-LBound(MatriceXY,1)+1) * (Ubound(MatriceXY,2) ...  
... -LBound(MatriceXY,2)+1)
```

Tableau dynamique avec redimensionnement

```
Dim Karnaugh() as boolean  
ReDim Karnaugh(20)
```

attention, les anciennes valeurs sont effacées (utiliser `ReDim Preserve`)
le type ne peut pas être modifié (sauf dans le cas d'un tableau de variant)

Le type **Array** permet de créer une liste de valeurs [de type tableau de variant]

```
MonTableau = Array("a", "b", "c")
```


Variables

● Variant

Variant est le type de données attribué à toutes les variables qui ne sont pas explicitement déclarées comme étant d'un autre type (à l'aide d'instructions telles que Dim, Private, Public ou Static). Le type de données Variant ne possède aucun caractère de déclaration de type. Variant est un type de données spécial pouvant **contenir** des données de toutes sortes, à l'exception des données de type String de longueur fixe.

La valeur **Empty** désigne une variable de type Variant qui n'a pas été initialisée (c'est-à-dire à laquelle aucune valeur initiale n'a été affectée). Un variant peut aussi contenir la valeur **Null** ou **Error**

Lorsqu'un variant contenant une chaîne est utilisé dans une opération requérant un type numérique, le variant est automatiquement converti en nombre.

Exemple:

```
Dim MyVar As Variant
MyVar = "20"
Cells(1, 1) = MyVar + 10
```

La valeur affichée dans la cellule est 30

Variables

● Date (très important pour la gestion des données industrielles horodatées)

Les dates sont comprises entre le 1er janvier 1900 et le 31 décembre 9999, les heures allant de 0:00:00 à 23:59:59.

Le type date est stocké dans un *double*. La *partie entière* correspond au *nombre de jours écoulés* depuis le 1er Janvier 1900; la *partie fractionnaire* correspond à la fraction de jour écoulée (1h = 1/24 de jour, 1mn = 1/(24x60) de jour , 1s =1/(24x60x60) de jour).

Exemple :

```
Dim ma_date As Date
Dim mon_heure, ma_dateheure As Double
ma_date = "16/09/2008"
mon_heure = 0.5 'une demi-journée de plus soit 12H
ma_dateheure = ma_date + mon_heure
Cells(3, 1) = ma_dateheure
```

donne comme résultat 16/09/08 12:00 PM (dépend du format d'affichage choisi)

- la conversion de type est implicite depuis le format chaîne.
- utilisez la fonction IsDate pour savoir si une valeur *variant* peut être convertie en date ou en heure.
- VBA propose des fonctions de conversion (DateSerial , TimeSerial)

Variables

● Type utilisateur

Un type utilisateur définit une **structure** contenant les types de base.

Exemple:

Déclaration du type

```
Public Type FicheEtudiant
    Nom As String*30           'variable type String contenant le nom.
    DateNaissance As Date     'variable type Date contenant la date de naissance .
    Sexe As Boolean           'variable type Boolean définissant le sexe
                              (False=fém,True=masc.).
End Type
```

Usage du type :

```
Dim NewEtudiant as FicheEtudiant
NewEtudiant.Nom = "Dupont"
NewEtudiant.DateNaissance = "25/10/82"
NewEtudiant.Sexe = True
```

ou bien :

```
Dim NewEtudiant as FicheEtudiant
With NewEtudiant
    .Nom = "Dupont"
    .DateNaissance = "25/10/82"
    .Sexe = True
End With
```

Variables

● Type objet

Un objet [ou classe] est une variable structurée permettant de manipuler des concepts évolués. Les valeurs associées à un objet s'appellent des **paramètres**, les fonctions appliquées à un classe sont des **méthodes**

Pour pouvoir utiliser un objet, il faut commencer par le **déclarer** puis l'**instancier** :

```
Dim monObjet As MaClasse
Set monObjet = New MaClasse
monObjet.param1 = 10
monObjet.Afficher ...
```

Ou

```
Dim monObjet As New MaClasse
...
```

Lorsque l'objet n'est pas **instancié** mais seulement déclaré, il est alors équivalent à "Nothing" .

Destruction d'un objet `Set monObjet = Nothing`

Il existe de nombreux objets prédéfinis dans Excel : `WorkBook`, `Range`

Variables

- **Hiérarchie des principaux objets Excel**

Application

 Workbooks

 WorkBook

 Worksheets

 WorkSheet

 Range, Cells

Exemple de référencement :

```
Excel.Workbooks(1).Worksheets(1).Range("A1").Value = "salut"
```

L'objet Cell n'existe pas au sens strict en Excel

Référence : <http://rdorat.free.fr/Enseignement/VBA/VBA/ObjExcel.html>

Instructions de contrôle

- **If** (condition) **Then**
instruction 1
instruction 2
...
Else
instruction 1
...
End If
- **If** (condition) **Then**
instruction 1
...
Elseif (condition) **Then**
instruction 1
...
End If
- **For** (variable) = (début) **To** (fin)
instruction 1
...
Next [nom de la variable d'incrément]

Expression des conditions

- opérateur de comparaison
 $a = b$ $a \leq 10$ $a \neq 3$
`ma_chaine = "Bonjour"`
- opérateur logique
 $(a = b) \text{ AND } (a \geq 10)$
 $(a = b) \text{ OR } \text{NOT}(a \leq 10)$
 $(a \neq b) \text{ XOR } (a = c)$

Macros

- Une macro est une procédure dont l'exécution est lancée depuis :
 - le menu Outils/Macro/Exécuter
 - par un raccourci clavier
 - par un bouton de la barre d'outils formulaire
 - une image
 - un bouton de la barre d'outils
- ↪ Le code VBA d'une macro peut être généré automatiquement par l'enregistreur de macro

Procédures

- Une procédure commence par `Sub` et termine par `End Sub`. Elle exécute des actions mais ne renvoie pas de valeurs .

Exemple sans argument: procédure de recopie la cellule active 1 ligne vers le bas et 0 colonne vers la droite, puis sélection de cette nouvelle cellule

```
Sub Copie_Decale()  
    ActiveCell.Copy ActiveCell.Offset(1,0)  
    ActiveCell.Offset(1,0).Select  
End Sub
```

Exemple avec argument d'entrée: les références de la cellule à décaler sont passées en paramètre de type chaîne .

```
Sub Copie_Decale_bis(AdressePlage As String)  
    Dim Ma_Plage As Range  
    Set Ma_Plage = Range(AdressePlage)  
    Ma_Plage.Copy Plage.Offset(1,0)  
    Ma_Plage.Offset(1,0).Select  
End Sub
```

Pour exécuter cette procédure, taper `Copie_Decale_bis("C3")` dans la fenêtre **Exécution**

Fonctions

- Une fonction accepte un ou plusieurs arguments en entrée et retourne un argument . Elle commence par **Function** et se termine par **End Function**

Exemple: fonction retournant le code couleur de la cellule dont les coordonnées sont passées en paramètre de type *chaîne*. La fonction retourne une *chaîne* .

```
Public Function Code_Couleur(AdressePlage As String) As String
    Dim Ma_Plage As Range
    Dim Index as integer
    Set Ma_Plage = Range(AdressePlage)
    Index = Ma_Plage.Interior.ColorIndex
    Code_Couleur = "Couleur de la cellule " & AdressePlage & " : " &
        str(Index)
```

End Function

Pour exécuter cette fonction, taper `range("B3") = Code_Couleur("A3")` dans la fenêtre **Exécution**. Le résultat est donné dans la cellule B3 .

Procédures/Fonctions

- Passage de paramètres

Par valeur (appel par défaut): C'est la valeur de la variable qui est passée dans la fonction. Tout calcul sur cette valeur est **local** à la fonction et n'affecte pas la valeur dans la structure appelante.

```
a = 5
increment(a)
MsgBox a           'la valeur affichée est 5
...
Sub increment (ByVal var1 as Integer)
    var1 =var1 + 1  ' var1 prend la valeur 5
End Sub
```

Par référence : le paramètre passé est l'adresse de la variable. Il en résulte que toute modification de la valeur dans la fonction affecte la valeur de la variable dans la structure appelante

```
a = 5
increment(a)
MsgBox a           'la valeur affichée est 6
...
Sub increment (ByRef var1 as Integer)
    var1 =var1 + 1  ' var1 pointe sur a
End Sub
```

Exercice 3

- Faire le calcul de la variance d'un tableau de données Excel et l'afficher dans une boîte de dialogue (voir suite transparents). L'interface de saisie est une feuille Excel

Le contenu du tableau est rangé à partir de la cellule A5 , le nb de lignes en B2 et le nombre de colonnes en B3.

1) Faire un programme basique de calcul en VBA comprenant la récupération des valeurs, le calcul puis l'affichage

2) Isoler le calcul dans une **fonction *STD(mon_tableau)*** dont le paramètre d'entrée est le seul nom de variable du tableau (utiliser un tableau dynamique, rechercher les dimensions du tableau d'entrée et appliquer le calcul)

| | A | B | C |
|---|-------------|----|----|
| 1 | Mon Tableau | A5 | |
| 2 | Nb lignes | | 3 |
| 3 | Nb colonnes | | 2 |
| 4 | | | |
| 5 | | 10 | 12 |
| 6 | | 10 | 12 |
| 7 | | 10 | 12 |
| 8 | | | |
| 9 | | | |

3) On souhaite que le tableau soit directement sélectionné à la souris. Utiliser les propriétés de l'objet **Selection** pour récupérer les données.

Autres Fonctionnalités

● Envoyer un message d'avertissement

```
Dim Mon Message as string
MonMessage = "Vous n'êtes pas autorisé
à mener cette action"+Chr$(13)+
"Opération annulée"
MsgBox MonMessage
```



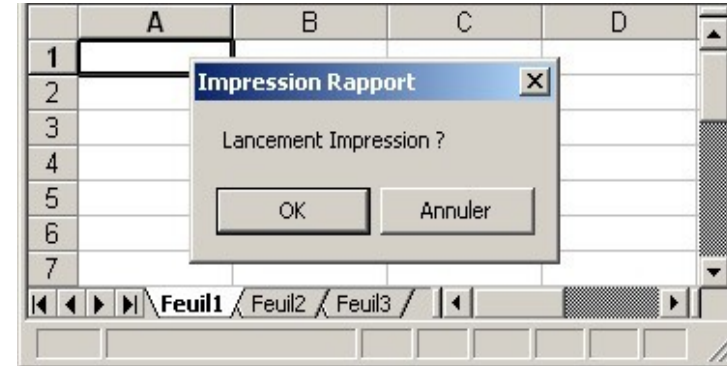
● Envoyer un message avec réponse type choix

```
Dim MonMessage, Titre As String
Dim Saisie As Integer

MonMessage = "Lancement Impression ?"
Titre = "Impression Rapport"

Saisie = MsgBox(MonMessage, vbOKCancel, Titre)

If Saisie = vbOK Then MsgBox"En cours"
```

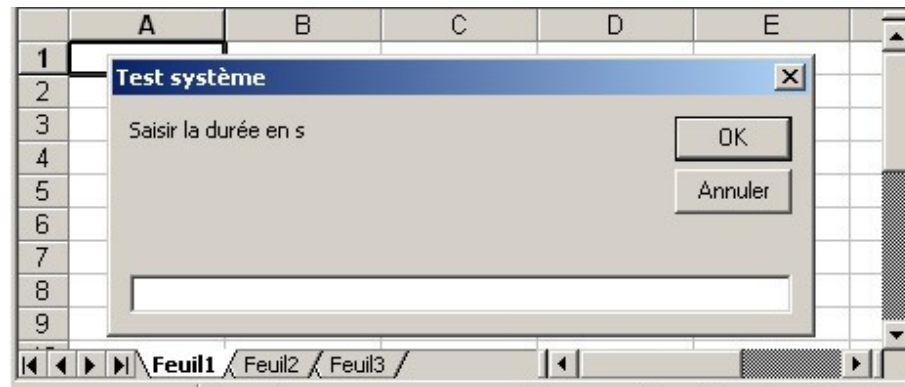


Autres actions

- Saisie d'une valeur

```
Dim Saisie as Variant
```

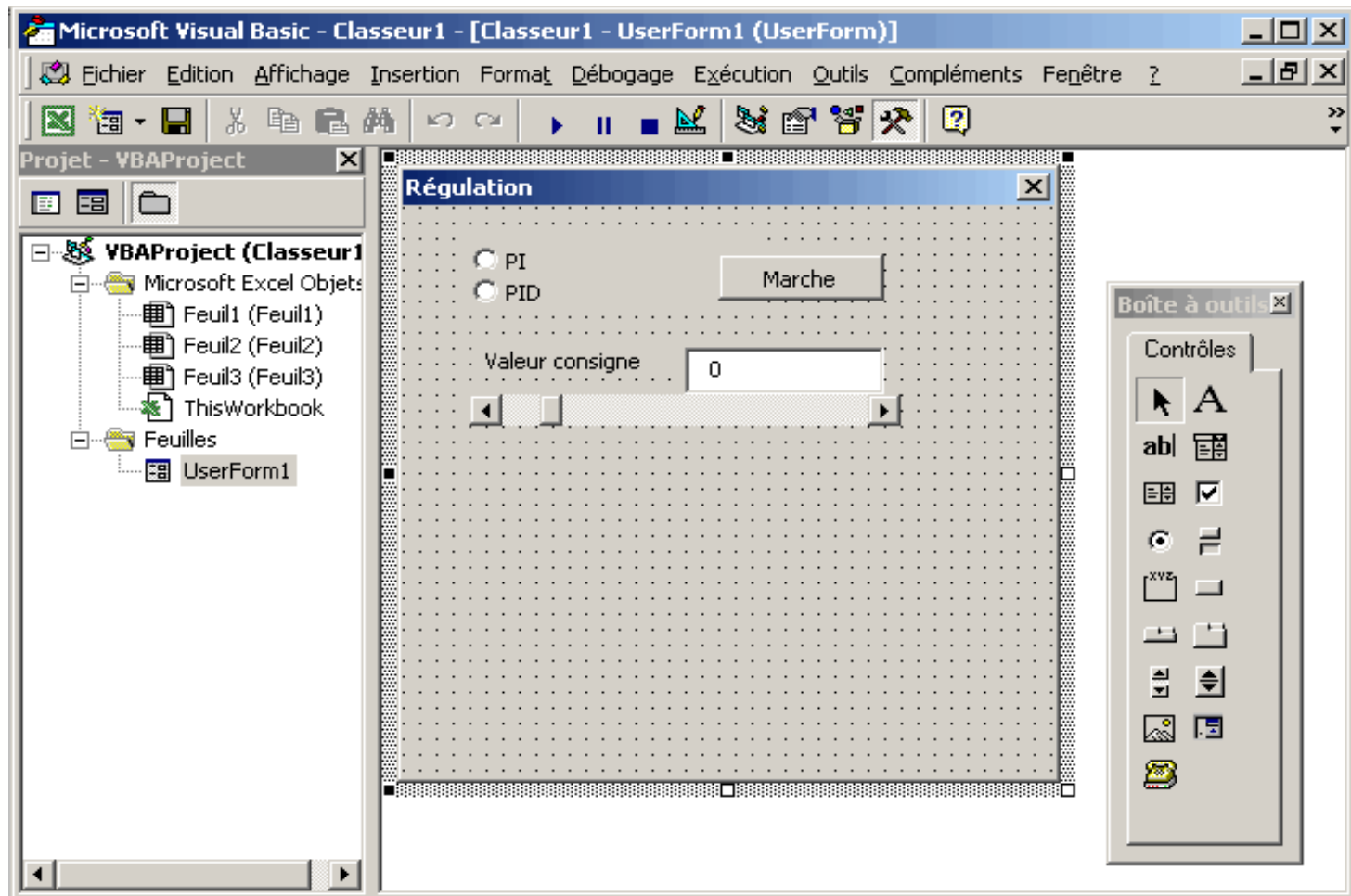
```
Saisie = InputBox("Saisir la durée en s", "test système")
```



ATTENTION : Les fonctions `MsgBox` et `InputBox` sont **bloquantes** jusqu'à réponse de l'opérateur.

Utilisation de UserForm

- permet de créer une fenêtre dans une application, indépendamment des feuilles Excel.
- En VB pur , la fenêtre est l'élément de base de l'IHM



Utilisation de UserForm

- les **contrôles** usuels (curseur, case à cocher, fenêtre texte...) peuvent être insérés dans un UserForm
- Par programmation, toutes les **propriétés** des contrôles sont accessibles, ainsi que les cellules des feuilles Excel (la valeur d'un contrôle peut être lié directement à celle d'une cellule)
- par défaut, le Userform n'est pas affiché. Pour le faire passer en premier plan, il faut utiliser la méthode :

```
My_UserForm. Show
```