eBook Gratuit

# APPRENEZ Visual Basic 6

eBook gratuit non affilié créé à partir des **contributeurs de Stack Overflow.**

#vb6

# Table des matières

# À propos

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: visual-basic-6

It is an unofficial and free Visual Basic 6 ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official Visual Basic 6.



Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

# Chapitre 1: Démarrer avec Visual Basic 6

## Remarques

Cette section fournit une vue d'ensemble de ce que vb6 est et pourquoi un développeur peut vouloir l'utiliser.

Il devrait également mentionner tous les grands sujets dans vb6, et établir un lien avec les sujets connexes. La documentation de vb6 étant nouvelle, vous devrez peut-être créer des versions initiales de ces rubriques connexes.

## Examples

### Installation ou configuration

Instructions détaillées sur la configuration ou l'installation de vb6.

### Bonjour le monde

```
' A "Hello, World!" program in Visual Basic.
Module Hello
  Sub Main()
      MsgBox("Hello, World!") ' Display message on computer screen.
  End Sub
End Module
```

Lire Démarrer avec Visual Basic 6 en ligne: https://riptutorial.com/fr/vb6/topic/3439/demarrer-avec-visual-basic-6

# Chapitre 2: Installation de VB6 sur Windows 10

## Examples

### Assistant d'installation

Assistant Visual Studio 6.0 Installer

Par défaut, les packages suivants ne s’installent pas correctement sous Windows 10:

- Visual Studio 6 Enterprise
- Visual Studio 6 Professional
- Visual Basic 6 Enterprise
- Visual Basic 6 Professional

Pour installer les packages ci-dessus, vous devez soit effectuer de nombreux ajustements et hacks de registre, soit utiliser le fantastique assistant Visual Studio 6.0 Installer de Giorgio Brausi.

Vous aurez besoin des éléments suivants avant de commencer:

- Vos CD et clés d'origine Visual Studio / Basic Program
- Vos CD MSDN d'origine
- Visual Studio Service Pack 6
- Assistant Visual Studio 6.0 Installer
- Sur Windows 10 version 1511 ou ultérieure, vous aurez besoin de droits d'administrateur.

L' Assistant vous guidera à travers les étapes nécessaires pour une installation réussie de Visual Basic 6.

Notez que l'installation des applications serveur n'est pas possible actuellement.

Lire Installation de VB6 sur Windows 10 en ligne:
https://riptutorial.com/fr/vb6/topic/4230/installation-de-vb6-sur-windows-10

# Chapitre 3: Les variables

## Examples

### Types de variables

Il existe différents types de variables à des fins différentes. Dans Visual Basic 6, les types de variables suivants sont disponibles:

- Tableau
- Booléen
- Octet
- Devise
- Rendez-vous amoureux
- Double
- Entier
- Longue
- Unique
- Chaîne
- Une variante

Vous déclarez une variable en utilisant le mot-clé `Dim` :

```
Dim RandomNumber As Integer
```

Si vous ne spécifiez pas de type de variable, la variable sera par défaut `Variant` :

```
Dim Foo
```

est équivalent à

```
Dim Foo As Variant
```

## Booléen

Boolean est le type de variable le plus simple car il ne peut contenir qu'une des deux valeurs: True ou False.

```
Foo = True
Bar = False
```

Les booléens peuvent être utilisés pour contrôler le flux de code:

```
Dim Foo as Boolean
Foo = True
```

```
If Foo Then
  MsgBox "True"
Else
  MsgBox "False"
End If
```

## Entier

Un entier est un type de données numérique et peut contenir une valeur signée 16 bits (-32768 à +32767). Si vous savez qu'une variable ne contiendra que des nombres entiers (tels que 9) et non des nombres fractionnaires (tels que 5.43), déclarez-la comme un type de données entier (ou long).

```
Dim RandomNumber As Integer
RandomNumber = 9
```

Les entiers sont couramment utilisés comme compteurs dans les boucles `For...Next` :

```
Dim Counter As Integer

For Counter = 0 to 2
  MsgBox Counter
Next Counter
```

Essayer d'attribuer une valeur inférieure à -32768 ou supérieure à 32767 à un nombre entier entraînera une erreur d'exécution:

```
Dim MyNumber As Integer
MyNumber = 40000  'Run-time error '6': Overflow
```

## Chaîne

Une variable chaîne peut contenir un texte vide, un caractère, un mot ou un texte de longueur variable. La valeur de chaîne doit être comprise entre guillemets ( `"` ).

```
Dim Fruit as String
Fruit = "Banana"
```

Si vous avez besoin de guillemets dans une chaîne littérale, vous utilisez deux guillemets ( `""` ).

```
Dim Quote as String
Quote = "Bill says: ""Learn VB!"""
```

Lire Les variables en ligne: https://riptutorial.com/fr/vb6/topic/7511/les-variables

# Chapitre 4: Procédures de fonction

## Introduction

Function est une série d'instructions entourées par des instructions "Function" et "End Function".

La fonction effectue une activité et renvoie le contrôle à l'appelant. Lorsqu'il renvoie le contrôle, il renvoie également une valeur au code d'appel.

Vous pouvez définir une fonction dans une classe, une structure et un module. Par défaut, c'est public. Cela signifie que vous pouvez l'appeler de n'importe où dans votre application qui a accès à la classe, à la structure ou au module dans lequel vous l'avez définie.

## Syntaxe

- [Modificateurs] Fonction Name_Of_The_Function [(Arg_List)] As Return_Type
- [Déclarations]
- Fonction de fin

## Remarques

- Les deux modificateurs de fonctions utilisés dans cet exemple sont Public & Private. Ces modificateurs définissent la portée de la fonction.
- Les fonctions avec une portée privée ne peuvent être appelées qu'à partir du fichier source à partir duquel elles ont été définies. Dans notre cas, il peut être appelé avec dans le module. Et ne peut pas être appelé en dehors du module.
- Les fonctions avec portée publique peuvent être appelées à la fois à l'extérieur et à l'intérieur du module. Nous pouvons simplement dire que "nous pouvons l'appeler n'importe où dans le programme".
- Le modificateur par défaut de la fonction est Public.
- Par défaut, les arguments de la fonction sont passés par référence (dans un sujet séparé, cela sera expliqué en détail).

## Examples

### Créer et appeler une fonction

Cet exemple utilisant un projet EXE standard avec ajout d'un fichier de module.

- Créer un nouveau projet "EXE standard". Donc, ici, un formulaire sera ajouté au projet par défaut.
- Ajouter un fichier de module au projet
- Placez un bouton de commande sur le formulaire
- Créer un bouton de commande Cliquez sur un événement.

### Code du module

Création de deux fonctions dans le module. L'une est une fonction publique (FnAdd). Il prend deux arguments Integer val_1 & val_2. Il retourne un entier. Cette fonction ajoute les deux arguments et renvoie la valeur à l'appelant. Avant l'addition, les deux arguments subissent un processus dans une autre fonction. Qui est une fonction privée. Caractéristiques / règles de la fonction publique et privée expliquées dans la section Remarques.

```
Public Function FnAdd(val_1 As Integer, val_2 As Integer) As Integer

'Calling private function
val_1 = FnMultiplyBy5(val_1)

'Calling private function
val_2 = FnMultiplyBy5(val_2)

'Function return statement
FnAdd = val_1 + val_2

End Function
```

Vous trouverez ci-dessous la fonction privée du module. Il faut un nombre entier d'arguments val. Il retourne un entier. Cette fonction multiplie la valeur 5 par l'argument et renvoie le résultat à l'appelant.

```
Private Function FnMultiplyBy5(Val As Integer) As Integer

'Function return statement
FnMultiplyBy5 = Val * 5

End Function
```

### Code de forme

Dans le bouton de commande, cliquez sur Evénement. Nous appelons ici la fonction publique du module "FnAdd"

```
Private Sub Command1_Click()
Debug.Print FnAdd(3, 7)
End Sub
```

### Résultat dans la fenêtre immédiate

```
50
```

Lire Procédures de fonction en ligne: https://riptutorial.com/fr/vb6/topic/9227/procedures-de-fonction

# Chapitre 5: Syntaxe de base

## Examples

### if / else déclaration

```
If condition Then
    code to execute if true
ElseIf condition Then
    code
Else
    code to execute if conditions are both false
End If
```

### pour la boucle

```
For I as Integer = 1 To 10 Step 1
    code to execute
Next
```

L'étape est facultative et l'étape 1 est la valeur par défaut. Step lui dit comment compter, donc -1 voudrait le soustraire 1 à chaque fois et l'étape 5 l'ajouterait 5 à chaque fois à travers la boucle.

Si la boucle doit être arrêtée, l'instruction `Exit For` peut être utilisée, comme dans l'exemple ci-dessous.

```
Dim iIndex as integer

For I as Integer = 1 To 10 Step 1

Debug.Print  I
iIndex = I * 10

If iIndex > 90 Then
    Exit For
End If

Loop
```

Ici, au lieu d'imprimer 1 à 10, il s'arrêtera à 9, car la condition demandait au processus de s'arrêter lorsque iIndex atteint 90.

### Do Loop

Un autre type courant de boucle dans Visual Basic est la `DO loop` , qui exécutera un morceau de code continuellement jusqu'à ce qu'il soit dit d'arrêter. Au contraire de certaines autres boucles par lesquelles des index sont utilisés pour arrêter le processus, dans cette boucle particulière, il convient de lui indiquer de s'arrêter.

Un exemple simple illustrant la boucle est le suivant

```
Dim iIndex1 As Integer
iIndex1 = 1

Do
   Debug.Print iIndex1
   iIndex1 = iIndex1 + 1

   If iIndex1 = 10 Then
      Exit Do
   End If
Loop
```

Le morceau de code ci-dessus prend un index, initialisé à 1, et l'incrémente. Un `Debug.Print` aidera à imprimer l'index pour compléter la boucle. Sur chaque boucle, le code vérifiera si l'index a atteint 10 et si et seulement si la condition est vraie, `Exit Do` sera exécuté, ce qui arrêtera la boucle.

## Sélectionner une déclaration de cas

```
    Dim number As Integer = 8
    Select Case number
        Case 1 To 5
            Debug.WriteLine("Between 1 and 5, inclusive")
            ' The following is the only Case clause that evaluates to True.
        Case 6, 7, 8
            Debug.WriteLine("Between 6 and 8, inclusive")
        Case 9 To 10
            Debug.WriteLine("Equal to 9 or 10")
        Case Else
            Debug.WriteLine("Not between 1 and 10, inclusive")
    End Select
```

Lire Syntaxe de base en ligne: https://riptutorial.com/fr/vb6/topic/9389/syntaxe-de-base

# Crédits

| S. No | Chapitres | Contributeurs |
|---|---|---|
| 1 | Démarrer avec Visual Basic 6 | Community, Scheffer |
| 2 | Installation de VB6 sur Windows 10 | ThunderFrame |
| 3 | Les variables | BlueEel |
| 4 | Procédures de fonction | Jeet |
| 5 | Syntaxe de base | Nadeem_MK, Talal Abdoh, user7491506 |