Débuter en Visual Basic 6

par bbil (bbil.developpez.com)

Date de publication : 19 janvier 2010

Dernière mise à jour : 23 août 2010

Vous trouverez dans ce document les bases pour débuter avec Visual Basic 6.

I - Présentation

Débuter en VB6 en 2010 semble être une hérésie. Cependant, au vue des nombreuses discussions ouvertes sur **les forums VB de dvp**, il m'a semblé utile d'écrire ce document, permettant d'orienter les nouveaux utilisateurs de Visual Basic 6.

II - VB6 sur developpez.com

Le site est formé d'un ensemble de rubriques, définies en fonction d'une technologie, un langage, un système d'exploitation...
La rubrique VB est l'une des plus anciennes de DVP, et englobe les différentes versions de Visual Basic, de la version 1 à la version 6 (1).
Suite à l'apparition du .Net, les versions ultérieures de VB - VB7 (VS2003), VB8 (VS2005), VB9 (VS2008),... - basées sur ce ".Net Framework" (2) ont été intégrées dans une nouvelle rubrique indépendante : **la Rubrique DotNet**. Je vous invite donc à vous tourner vers cette rubrique, et plus particulièrement **la sous-rubrique VBNet**, pour tout ce qui concerne les versions de Visual basic ultérieures à la version 6.

Un "sous-ensemble" de VB6, nommé par Microsoft "Visual Basic pour Application" ou VBA (3), définit le langage intégré dans diverses applications, dont principalement les applications de la suite Office de Microsoft,
mais aussi d'autres telles que Autocad, Catia ... Sur developpez.com le VBA est géré par une rubrique plus orientée bureautique : **Office** et ses "sous-rubriques" telles que : **Excel**, **Word**, **Outlook**, ou **PowerPoint**.

III - La rubrique VB

La rubrique VB comporte plusieurs pages destinées à vous aider pour l'apprentissage du langage VB.

- L'index VB ou page d'accueil: **http://vb.developpez.com/** propose les actualités de la rubrique (news, annonces d'articles..)
- Les tutoriels VB6 : **http://vb.developpez.com/cours/**regroupent les articles écrits par les membres de la rédaction
- La Faq VB6 : **http://vb.developpez.com/faq/** comporte un ensemble de réponses à vos questions élaborées à partir des réponses piochées dans les forums VB.
- La page Source VB6 : **http://vb.developpez.com/sources/** présente un ensemble de sources fournies par les "contributeurs" des forums VB.
- La page Outils VB6 :**http://vb.developpez.com/outils/** offre une sélection d'outils gratuits pour VB6.
- La page Livres VB6 : **http://vb.developpez.com/livres/** expose quelques critiques de livres effectuées par les membres de la rédaction.
- Les pages contributeurs : **Pages Contributeurs VB** reprennent les productions des "gros" contributeurs des forums VB6.
- Le Wiki de developpez.com : **le wiki de developpez.com** définit et explique les outils et les termes utilisés en VB6. Vous pouvez y contribuer en ajoutant ou modifiant des entrées.
- Les forum VB6 : **VB 6 et antérieur** permettent de consulter les questions/réponses ou de poser ses propres questions.

La rubrique VB a aussi en charge le langage VBScript (ou VBS). Voici quelques pages VBS :

- La FAQ VBS : **http://vb.developpez.com/faqvbs/** reprend un ensemble de réponses extraites du forum VBscript

(1) Voir Historiques des versions VB **http://wiki.developpez.com/Visual_Basic_6**
(2) **Qu'est-ce que le .NET Framework ?**
(3) **Comment savoir si l'on utilise VBA (Visual basic pour application) ou VB6 ?**

- Les tutoriels VBScript : **Pages cours VBScript** regroupent quelques articles sur VBscript écrit par les membres de la rédaction.
- Les forums VBScript : **VBScript** permettent la consultation ou la création de discussions.

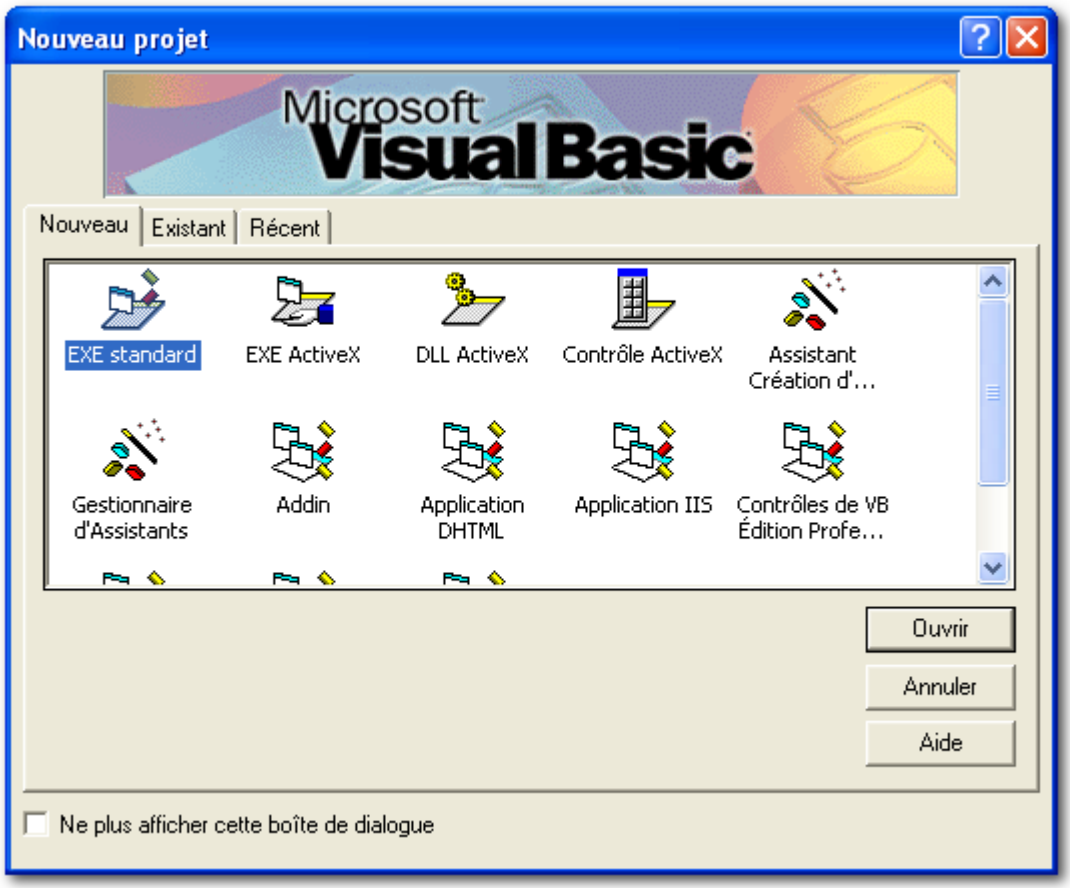
IV - Télécharger Visual Basic 6.0

Le logiciel Visual Basic 6.0, **n'est plus disponible auprès de Microsoft** depuis le 27/01/1999. Cependant, l'édition professionnelle du produit est disponible en téléchargement pour les abonnés MSDN.
En outre, une personne qui a acquis un logiciel Microsoft légalement peut procéder à sa revente.
Toutefois, tout transfert de licence, que ce soit par le biais d'une cession, d'un don ou d'un cadeau, doit être accompagné de la documentation et des manuels du produit, des disques originaux et des licences.

Des conditions de transfert supplémentaires peuvent être contenues dans votre CLUF (4). Je vous invite donc à la consulter attentivement avant tout transfert de licence.

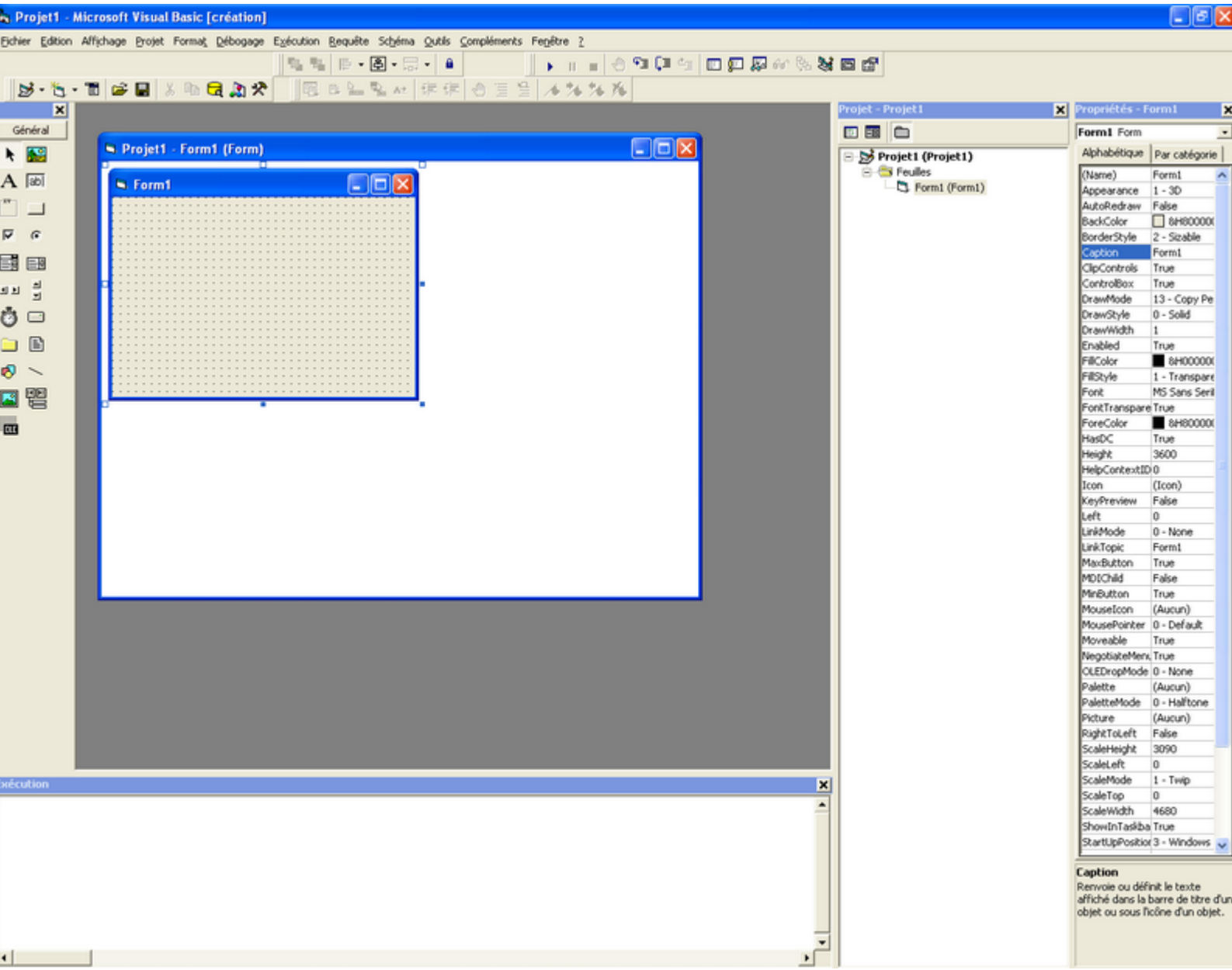
V - Création du projet

Pour commencer, nous allons créer la classique application "Hello world".
Au lancement de VB6, la fenêtre de sélection de type de projet est affichée :

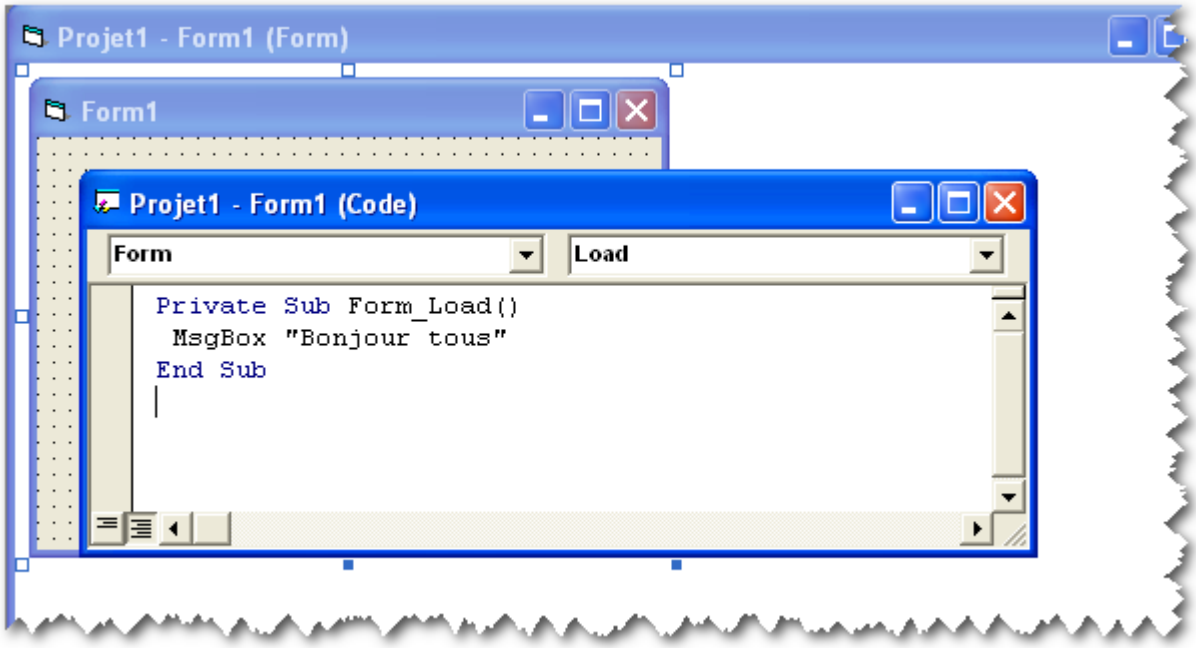


Dans ce document nous nous limiterons au projet standard, sélectionnez donc "Exe standard", ce qui permet de créer une première fenêtre "Form1".

(4) CLUF : Contrat de Licence Utilisateur Final



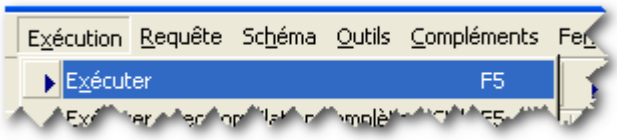
Modifiez ensuite le code de cette fenêtre, pour accéder au code qui lui est associé, vous pouvez "double-cliquer" sur celle-ci ou utiliser le menu "Affichage/Code"



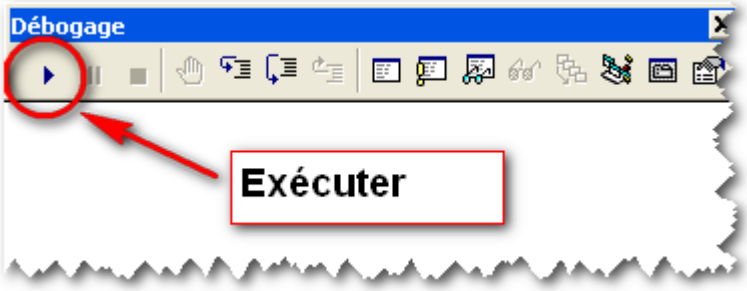

```
Private Sub Form_Load()
 MsgBox "Bonjour tous"
End Sub
```

VI - Exécution du programme

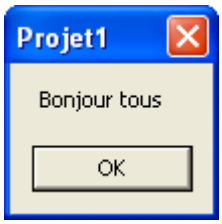
Une fois ce code saisi, utilisez la commande Exécuter, pour lancer le projet. Vous trouverez cette commande dans le menu Exécution :



soit en actionnant la Touche F5, ou le bouton Exécuter de la boîte à outils "Débogage" :

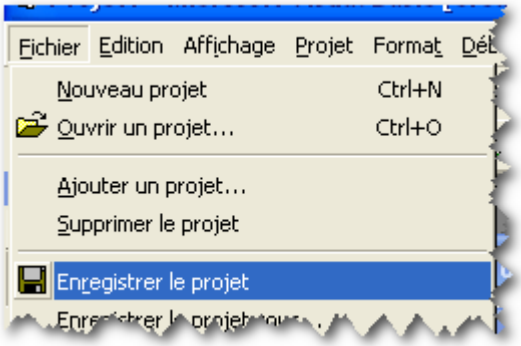


Le message désiré est alors affiché :

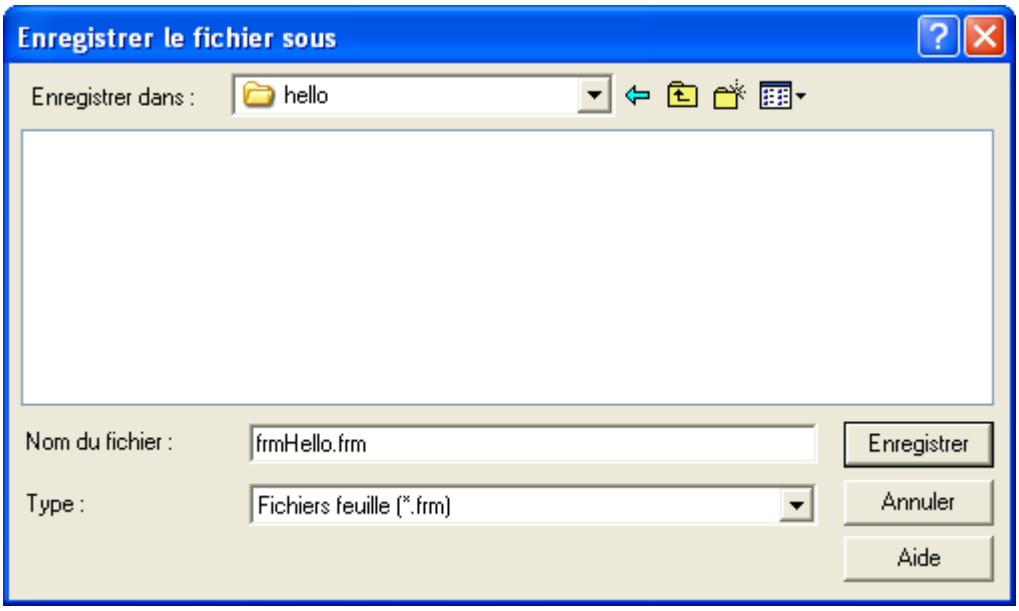


VII - Enregistrement du programme

Passez ensuite à l'enregistrement du programme, pour cela sélectionnez l'entrée : "Enregistrer le projet" du menu Fichier :



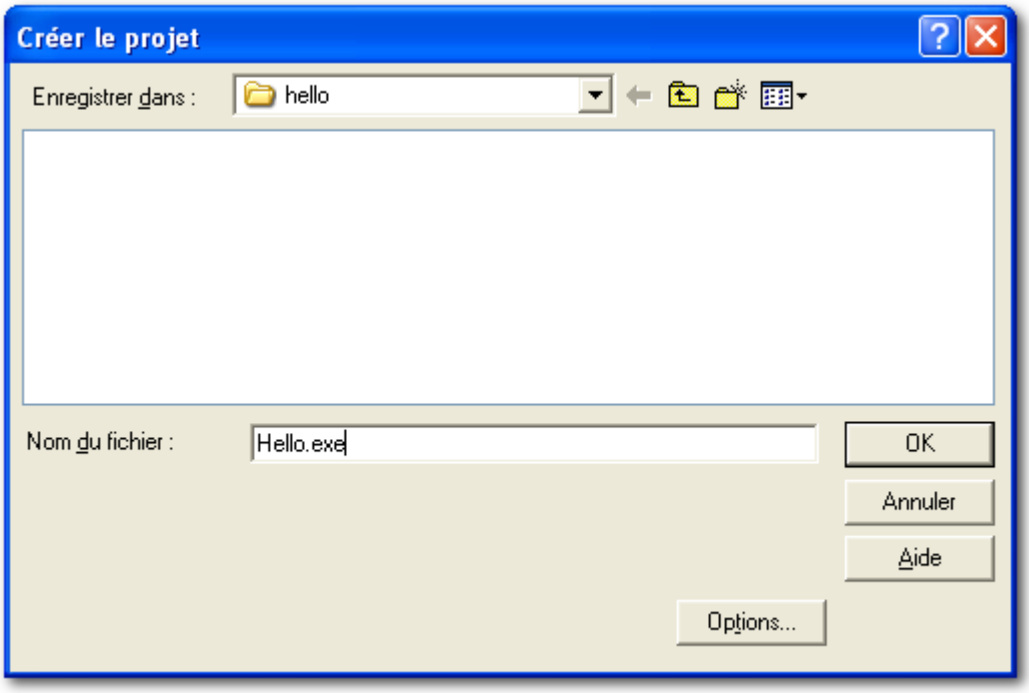
Ensuite sélectionnez le répertoire et le nom de fichier souhaités, grâce à la fenêtre "Enregistrer sous" qui vous permet en outre de créer le répertoire de destination :



Cette fenêtre est affichée pour tous les fichiers du projet.

VIII - Compilation du programme

Pour finir vous pouvez compiler le projet, afin qu'il soit exécutable en dehors de l'éditeur Visual Basic. Dans le menu fichier, sélectionnez "Créer prjHello.exe" (prjHello étant le nom du projet), si nécessaire changer le nom du fichier.



IX - Télécharger l'application Hello

Les sources VB6 du projet prjHello

X - Quelques liens vers des articles connexes

Démarrage de Visual Basic
L'environnement de développement
Bien implémenter un projet avec Visual Basic 6.0

XI - Présentation

Une variable permet de donner un nom significatif à un espace mémoire utilisé ensuite dans votre programme.

XII - Type de données

L'aide en ligne de VB6 vous permet de retrouver facilement les divers types de données disponibles. Vous pouvez aussi consulter sur ce site divers tutoriels traitant du sujet :

Les éléments de programmation 'Les Variables'

Descriptif des conventions typographiques du code Visual Basic

XIII - Déclaration de variables

Permissivité de VB6 vis à vis des variables

VB6 est très permissif et par défaut n'oblige pas à déclarer les variables.
Ainsi le simple fait d'utiliser une variable permet de la créer. La ligne d'instruction : MaVariable = "Une variable chaine" permet donc de créer une variable avec ce nom très explicite "MaVariable", de lui affecter la chaine entre guillemet comme valeur, et donc le type de la variable créée sera "String". Cependant la variable n'ayant pas été déclarée, son type peu être changé lors d'une autre affectation.
Pour vérifier cela la fonction TypeName permet d'afficher le type utilisé par la variable.

Code	Résultat (fenêtre exécution CTRL+G)

`Sub AfficheType()`
`  MaVariable = "Une variable chaine"`
`  Debug.Print " 1 : " & TypeName(MaVariable) _`
`  & " = " & MaVariable`
`  MaVariable = 123`
`  Debug.Print " 2 : " & TypeName(MaVariable) _`
`  & " = " & MaVariable`
`End Sub`
	1 : String = Une variable chaine
	2 : Integer = 123

En quoi le type de variable est-t'il important ?

Code	Résultat (fenêtre exécution CTRL+G)

`Sub ImportanceDuType()`
`  Var1 = 1 'Ici la var1 est un entier`
`  Var2 = 2 'Ici la var2 est un entier`
 	
`  Debug.Print "Opérateur + appliqué à 2 entier : ";`
`  Debug.Print "Var1 + Var2 =";`
`  Debug.Print Var1 + Var2`
`  Var1 = "1" 'Ici la var1 est une chaine ...`
`  Var2 = "2" 'Ici la var2 est une chaine`
 	
`  Debug.Print "Opérateur + appliqué à 2 chaine : Var1 + Var2 =";`
`  Debug.Print Var1 + Var2`
`End Sub`
	Opérateur + appliqué à 2 entier : Var1 + Var2 = 3
	Opérateur + appliqué à 2 chaine : Var1 + Var2 =12

Ce petit code nous montre un des problèmes pouvant être entrainés par un oubli de la déclaration de variable. Cette déclaration s'effectue grâce au mot clef Dim :

```
Dim stVar1 as string
Dim stVar2 as string
```

```
Dim iVar3 as integer
Dim iVar4 as integer
```

On remarque la répétition du type pour chacune des variables, un raccourci souvent utilisé par les débutants :

```
Dim stVar1,stVar2 as string
Dim iVar3, iVar4 as integer
```

ne donne pas le résultat attendu comme le montre le code suivant :

Code	Résultat (fenêtre exécution CTRL+G)

`Sub DeclareVariable()`
`  Dim stVar1,`
`  stVar2 As String 'Seule stVar2 est une String`
`  Dim iVar3, iVar4 As`
`  Integer 'Seul iVar4 est un entier`
`  Debug.Print "stVar1 : " & TypeName(stVar1)`
`  Debug.Print "stVar2 : " & TypeName(stVar2)`
`  Debug.Print "iVar3 : " & TypeName(iVar3)`
`  Debug.Print "iVar4 : " & TypeName(iVar4)`
`End Sub`
	stVar1 : Empty
	stVar2 : String
	iVar3 : Empty
	iVar4 : Integer

Option Explicit

En rendant obligatoire la déclaration des variables "l'option explicit" d'éviter bien des erreurs de frappe et d'appréciation de portées de variables.
Quelques petits exemples d'erreurs relevées par option explicit :

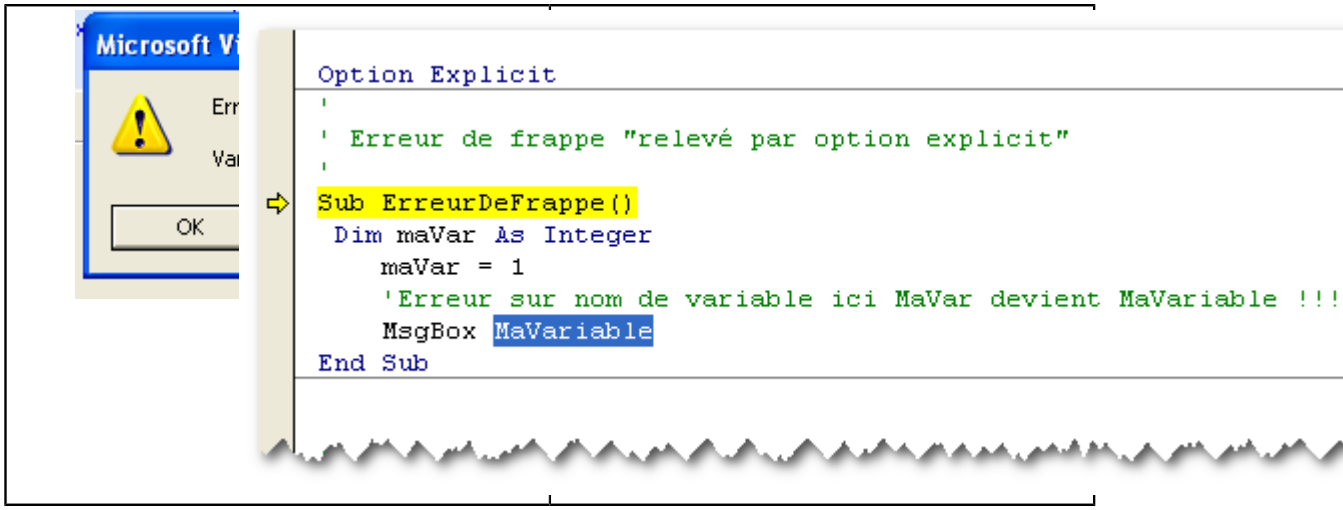
Erreur de frappe

Voici un code où une erreur est survenue lors de la saisie du nom d'une variable qui passe de maVar à MaVariable

```
Sub ErreurDeFrappe()
 Dim maVar As Integer
    maVar = 1
    'Erreur sur nom de variable ici MaVar devient MaVariable !!!
    MsgBox MaVariable
End Sub
```

Si on exécute ce code sans utiliser Option Explicit, VB ne signale aucune erreur, la fenêtre message box s'affiche, sans le contenu de la variable MaVar.

Par opposition, après l'ajout de l'option Explicit, l'erreur est signalée lors de l'exécution dans l'IDE Visual basic ou lors de la compilation.



Après action sur le bouton OK de la message box, le code est affiché signalant par une flèche jaune la ligne en cours d'exécution et surlignant en bleu le code en erreur.

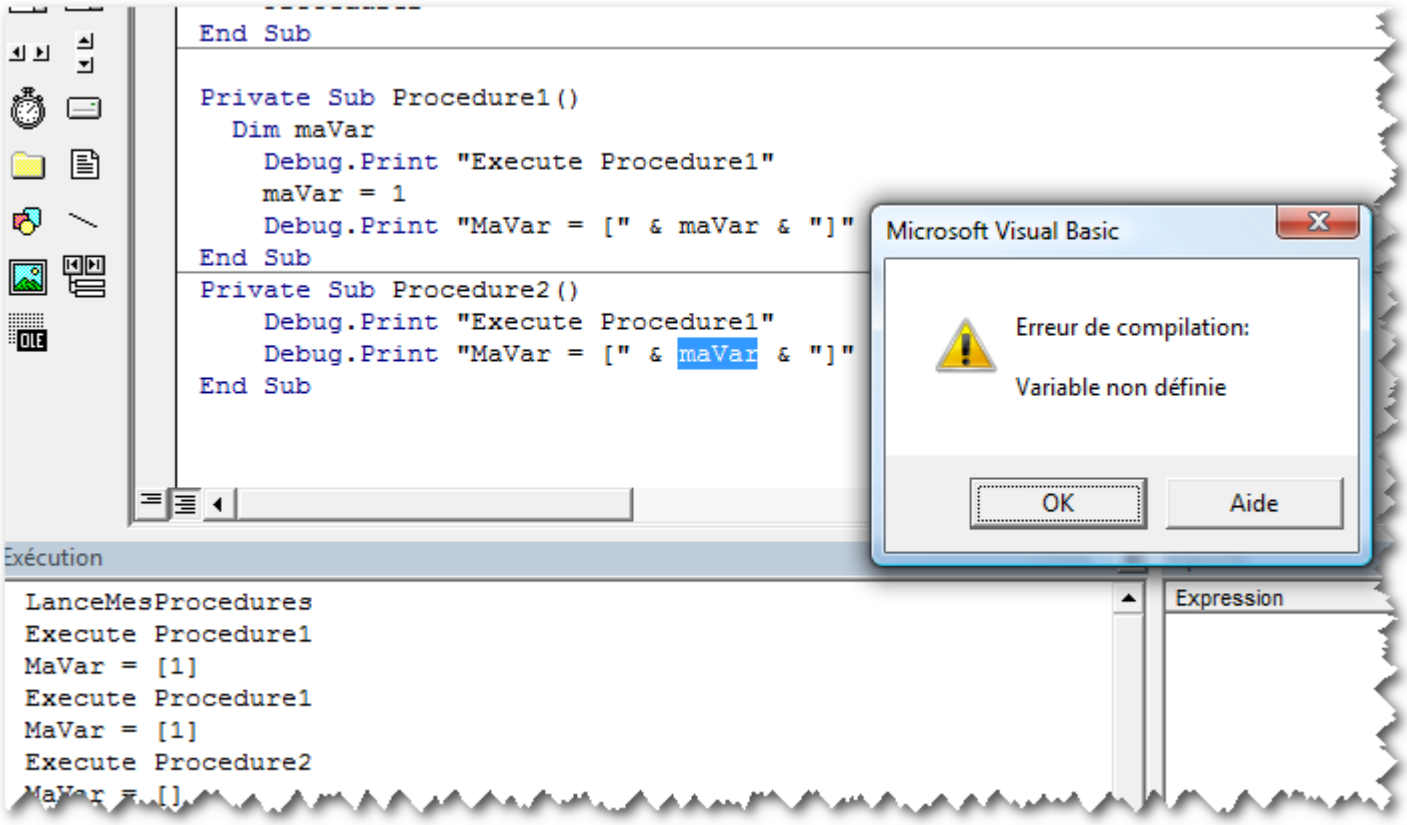
Erreur de portée

Une variable déclarée au mauvais endroit peut provoquer des erreurs de portée, ainsi une variable déclarée dans une procédure, n'est pas accessible dans une autre.

Code	Résultat (fenêtre exécution CTRL+G)

`Sub LanceMesProcedures()`
`  Procedure1`
`  Procedure2`
`End Sub`
`Private Sub Procedure1()`
`  Dim maVar`
`  Debug.Print "Execute Procedure1"`
`  maVar = 1`
`  Debug.Print "MaVar = [" & maVar & "]"`
`End Sub`
`Private Sub Procedure2()`
`  Debug.Print "Execute Procedure2"`
`  Debug.Print "MaVar = [" & maVar & "]"`
`End Sub`
	LanceMesProcedures
	Execute Procedure1
	MaVar = [1]
	Execute Procedure2
	MaVar = []

Si on exécute ce code sans utiliser Option Explicit, VB ne signale aucune erreur, mais la variable MaVar n'étant pas accessible dans la deuxième procédure, la valeur affichée lors de l'exécution de procédure2 est nulle, car il s'agit d'une "autre" variable créée automatiquement par VB.
Cependant, après l'ajout de Option Explicit, l'erreur est signalée lors de l'exécution dans l'IDE Visual basic ou lors de la compilation, la fenêtre



suivante est affichée :

Dans l'éditeur VBE la variable MaVar utilisée dans la deuxième procédure est surlignée en bleu.

Pour rendre ce code valide et faire en sorte que la variable MaVar soit accessible par les deux procédures, on peut déclarer en "global" cette variable en plaçant la ligne de déclaration en dehors de toute procédure, tout en haut de la feuille de code :

Code	Résultat (fenêtre exécution CTRL+G)

`Option Explicit`
`Dim maVar`
`'(...)`
`Sub LanceMesProcedures()`
`  Procedure1`
`  Procedure2`
`End Sub`
`Private Sub Procedure1()`
`  Dim maVar`
`  Debug.Print "Execute Procedure1"`
`  maVar = 1`
`  Debug.Print "MaVar = [" & maVar & "]"`
`End Sub`
`Private Sub Procedure2()`
`  Debug.Print "Execute Procedure2"`
`  Debug.Print "MaVar = [" & maVar & "]"`
`End Sub`
	LanceMesProcedures
	Execute Procedure1
	MaVar = [1]
	Execute Procedure2
	MaVar = [1]

XIV - Les tableaux de variables

XIV-A - Introduction aux tableaux

Les tableaux permettent de regrouper plusieurs "valeurs" au sein d'une même variable, ils sont particulièrement indiqués pour les traitements "en boucle" qui permettent d'accéder aux divers éléments du tableaux grâce à une variable numérique servant d'index.

On distingue essentiellement deux sortes de tableaux :

1 Taille fixe : le nombre d'éléments du tableau est défini au moment de la déclaration et ne peut être modifié ensuite ;
2 Taille dynamique : le nombre d'éléments du tableau n'est pas défini au moment de la déclaration, mais ensuite grâce à une instruction spécifique, ce nombre d'éléments peut évoluer en fonction des traitements du programme.

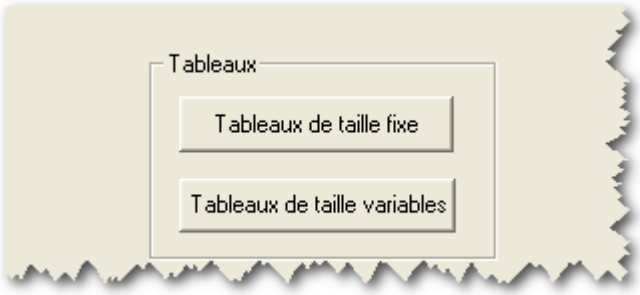
XIV-B - Tableaux de taille fixe

La déclaration du tableau s'effectue avec l'instruction Dim sous la forme :
Dim tb(*IndexBas* To *IndexHaut*) as *Type*
L'index bas étant facultatif, lorsqu'il est omis il prend la valeur par défaut 0 et dans ce cas :
Dim tb(*0* To *IndexHaut*) as *Type*
peut s'écrire de façon simplifiée ainsi :
Dim tb(*IndexHaut*) as *Type*

La valeur par défaut de l'index bas habituellement égale à 0 peut-être modifiée et passée à 1 par l'instruction suivante en début du module :

```
Option Base 1
```

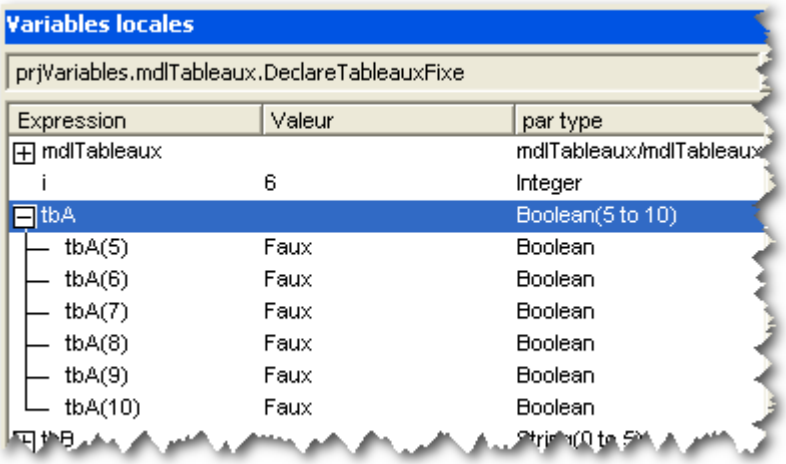
L'application prjVariables en téléchargement avec cet article permet de mieux appréhender l'utilisation des variables tableaux :



utilisez pour cela le bouton "Tableaux de taille fixe"
L'instruction stop présente dans le code permet de stopper l'exécution du programme et de visualiser le code de déclaration, mais aussi grâce à la fenêtre "Variables locales" (accessible par le menu affichage de l'éditeur Visual basic) d'observer les variables.
Ainsi la déclaration :

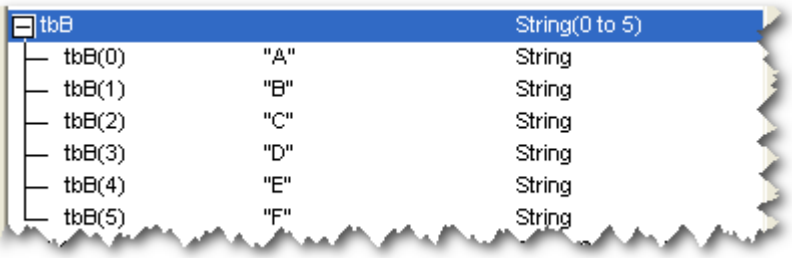
```
' tableau une dimension bornes fixées
    Dim tbA(5 To 10) As Boolean
```

permet de déclarer une variable tableau nommée tbA contenant 6 éléments de type booléen avec un index variant de 5 à 10.



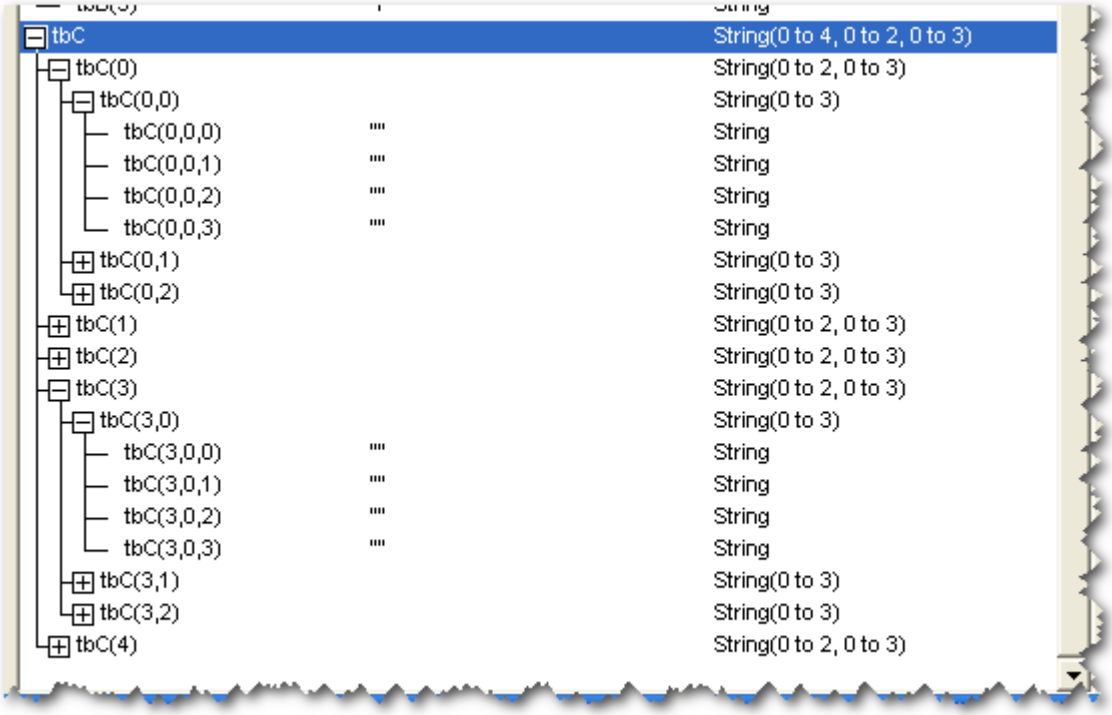
La variable tableau tbB met en évidence la déclaration en mode "court" avec index bas à 0 :

```
    'Tableau une dimension utilisant la valeur par défaut pour la borne basse
    Dim tbB(5) As String
```



Pour finir, le dernier tableau tbC nous montre que les tableaux ne sont pas limités à une seule dimension :

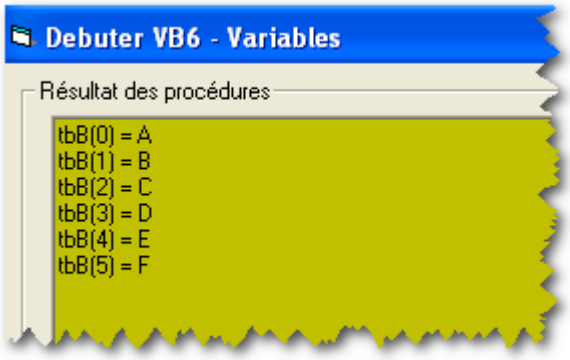
```
    'Tableau multi-dimensionnel
    Dim tbC(4, 2, 3) As String
```



Un appui sur la touche F5 permet de faire "repartir" le programme exécutant la boucle d'affichage suivante :

```
'pour finir par une boucle montrant l'utilisation en lecture d'un tableau
For i = LBound(tbB) To UBound(tbB)
    frmVariables.AjtLog "tbB(" & i & ") = " & tbB(i)
Next
```

dont le résultat est affiché dans le textbox présent sur la boîte de dialogue :



XIV-C - Tableaux de taille variable

Par opposition aux tableaux de taille fixe, les tableaux de taille variable peuvent avoir leur taille ajustée en fonction du nombre d'éléments à stocker.
La déclaration d'un tableau de taille variable se fait de la même manière que pour les tableaux de taille fixe grâce à l'instruction Dim, cependant les tailles des diverses dimensions ne sont pas définies par l'instruction Dim mais doivent l'être lors de l'exécution du programme par une instruction ReDim.

La déclaration de la variable tableau s'effectue grâce à une instruction Dim :

```
Dim tbB() As String
```

Ensuite, une instruction Redim permet de spécifier la taille de la variable :

```
ReDim tbB(6)
```

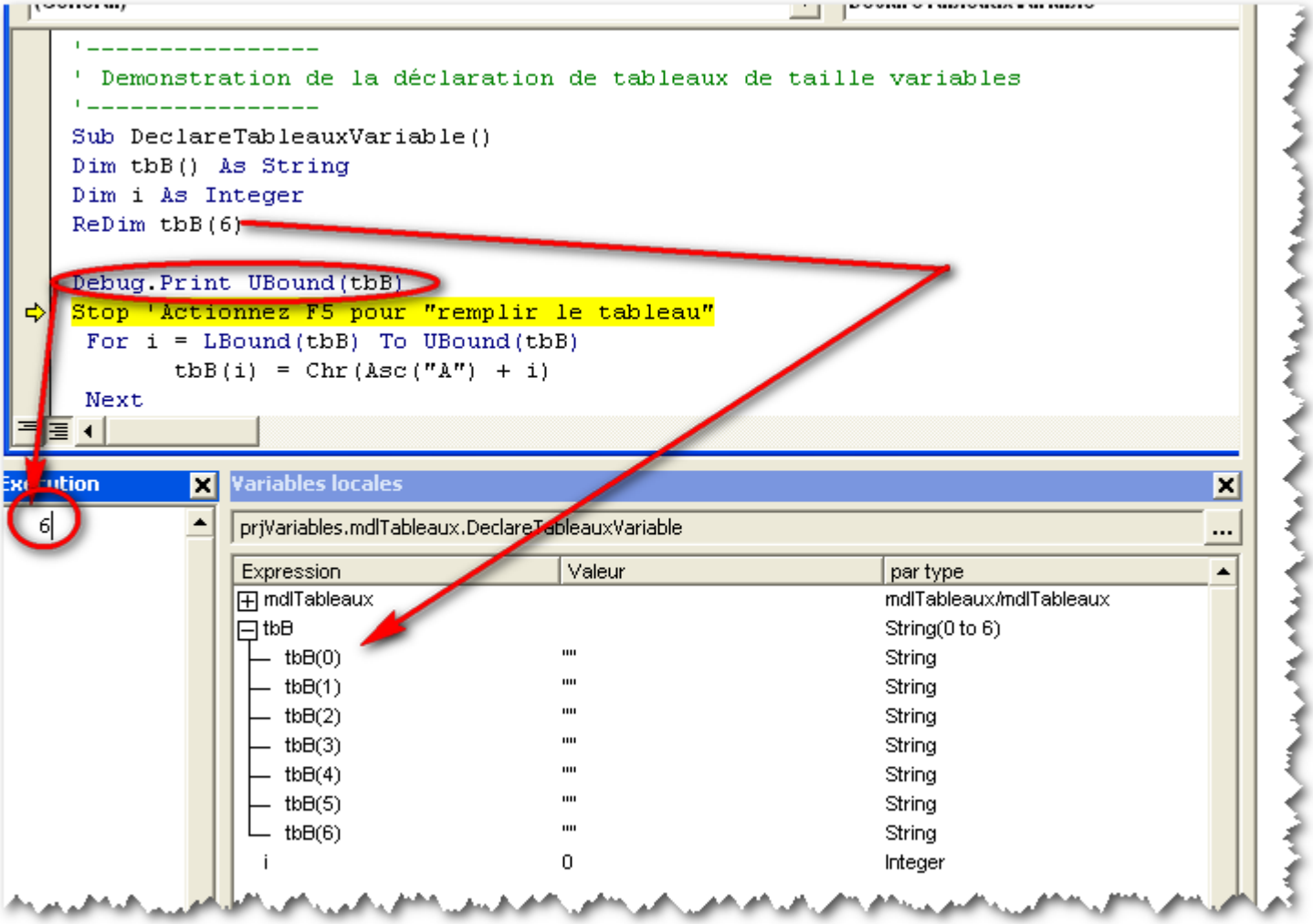
Pour visualiser ce fonctionnement utiliser le bouton "Tableau de taille variable" dans l'application exemple lancée dans l'éditeur VB.

Au premier arrêt du code, la fenêtre exécution (CTRL+G) permet d'afficher le résultat de l'instruction :

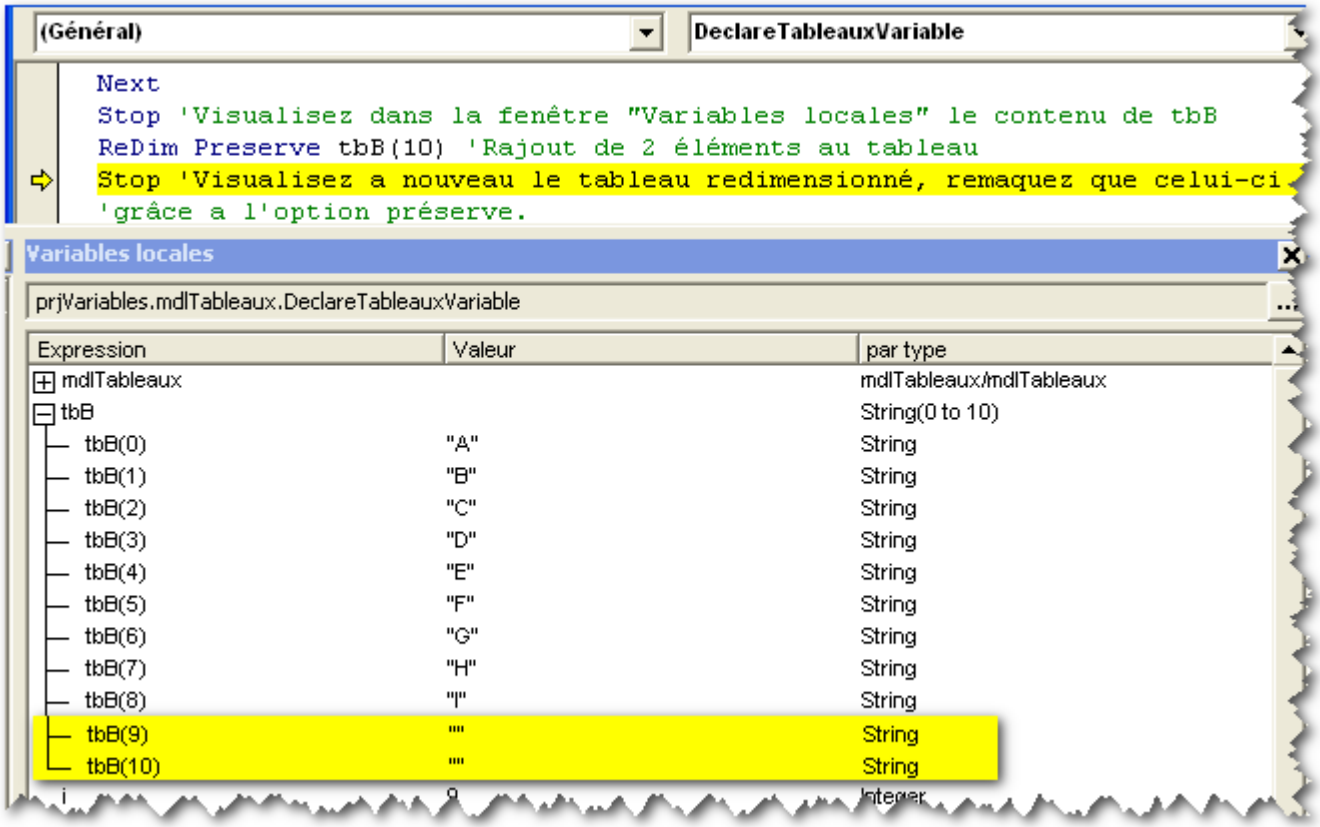
```
Debug.Print UBound(tbB)
```

et donc l'index du dernier élément du tableau (ici 6). Par ailleurs la fenêtre "Variable locales" permet d'afficher les 7 éléments composants le tableau.

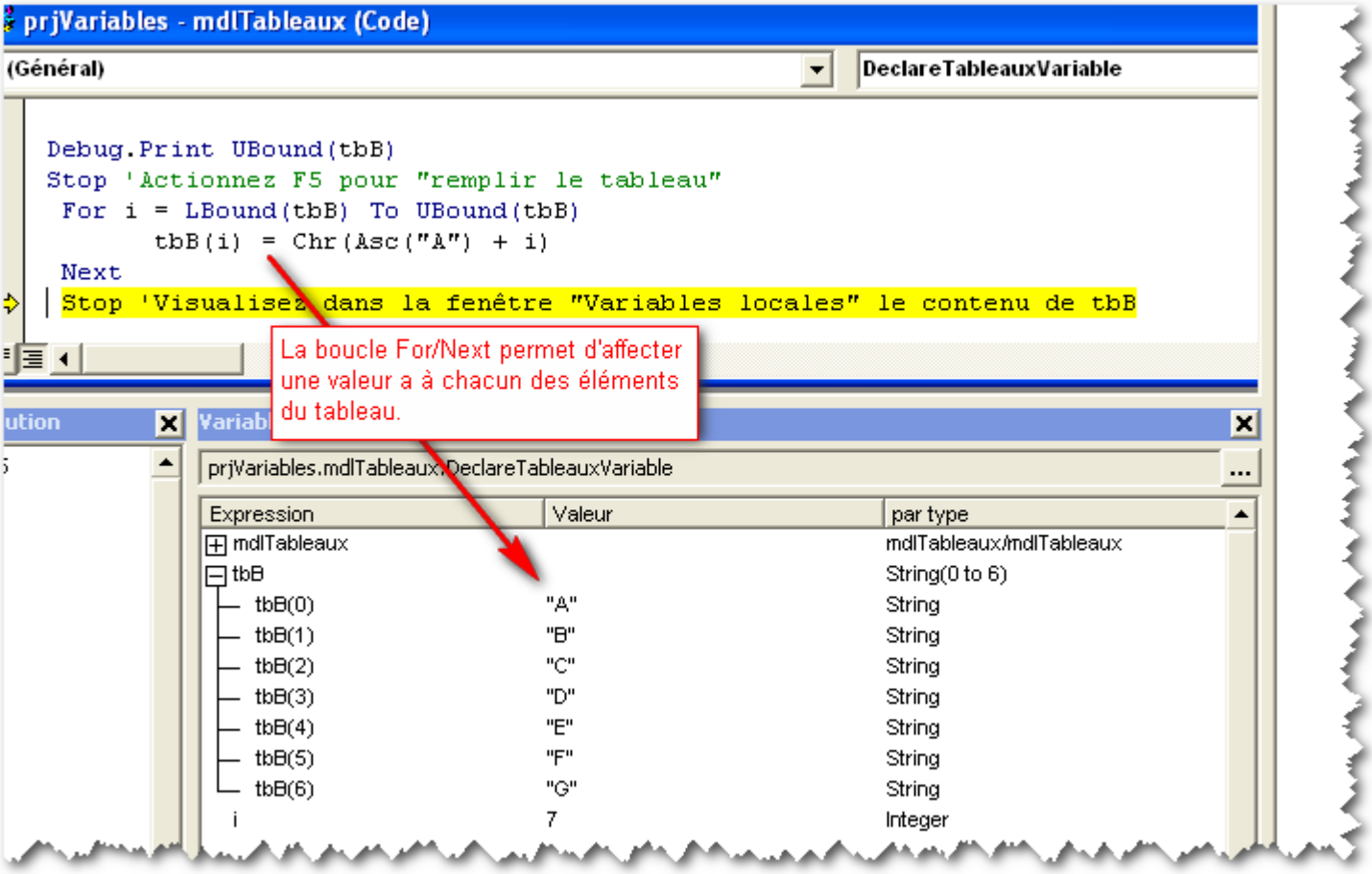
```
Sub DeclareTableauxVariable()
 Dim tbB() As String
 Dim i As Integer
 ReDim tbB(6)
 Debug.Print UBound(tbB)
 Stop 'Actionnez F5 pour "remplir le tableau"
```



Un appui sur la touche F5 permet d'exécuter le code jusqu'à la prochaine instruction Stop, ensuite l'affectation des différents éléments du tableau est effectuée grâce à une boucle.



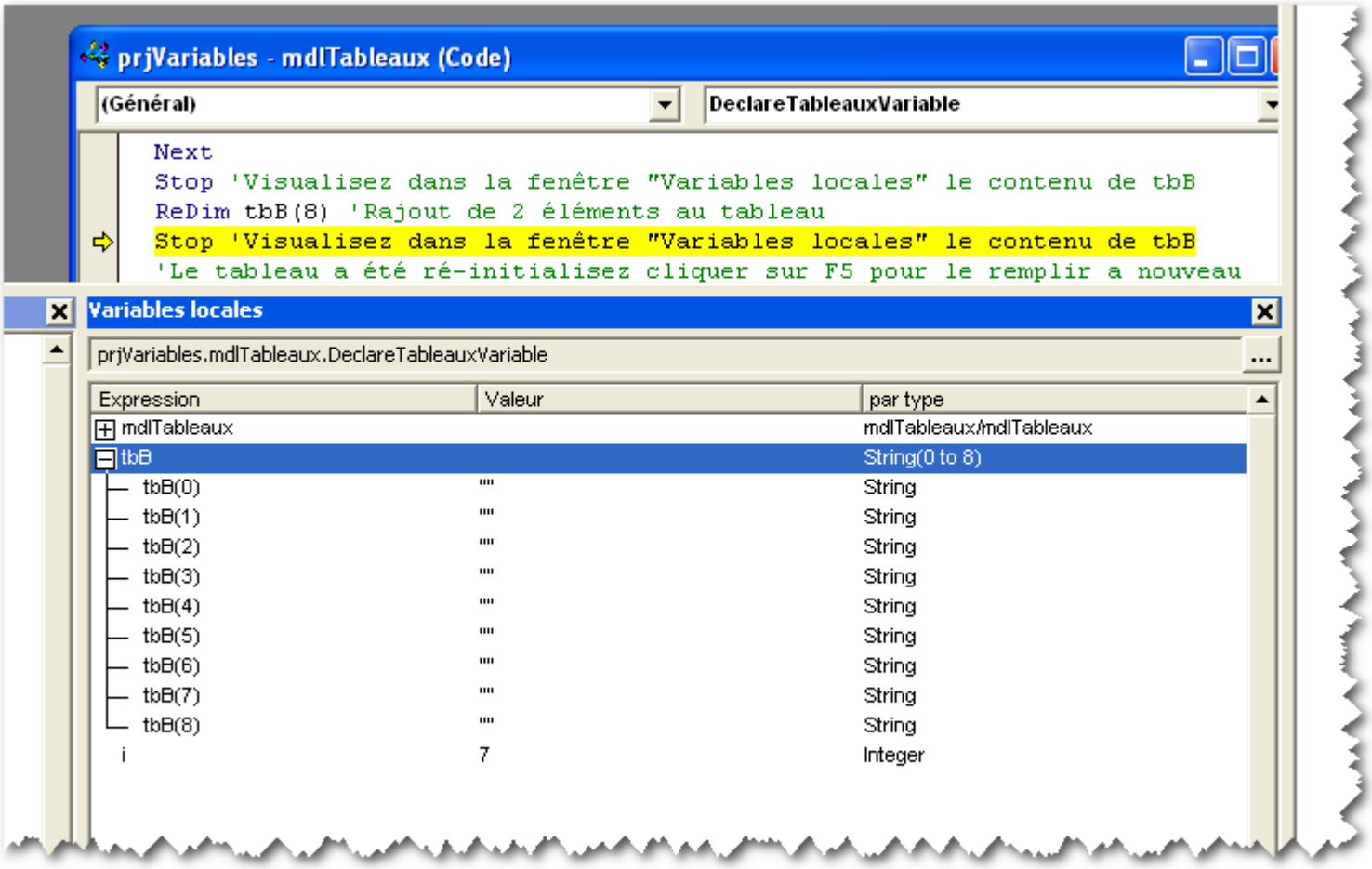

```
Stop 'Actionnez F5 pour "remplir le tableau"
 For i = LBound(tbB) To UBound(tbB)
       tbB(i) = Chr(Asc("A") + i)
 Next
 Stop 'Visualisez dans la fenêtre "Variables locales" le contenu de tbB
```



La suite du code montre le résultat de l'utilisation de l'instruction ReDim.

```
Stop 'Visualisez dans la fenêtre "Variables locales" le contenu de tbB
ReDim tbB(8) 'Rajout de 2 éléments au tableau
Stop 'Visualisez dans la fenêtre "Variables locales" le contenu de tbB
```

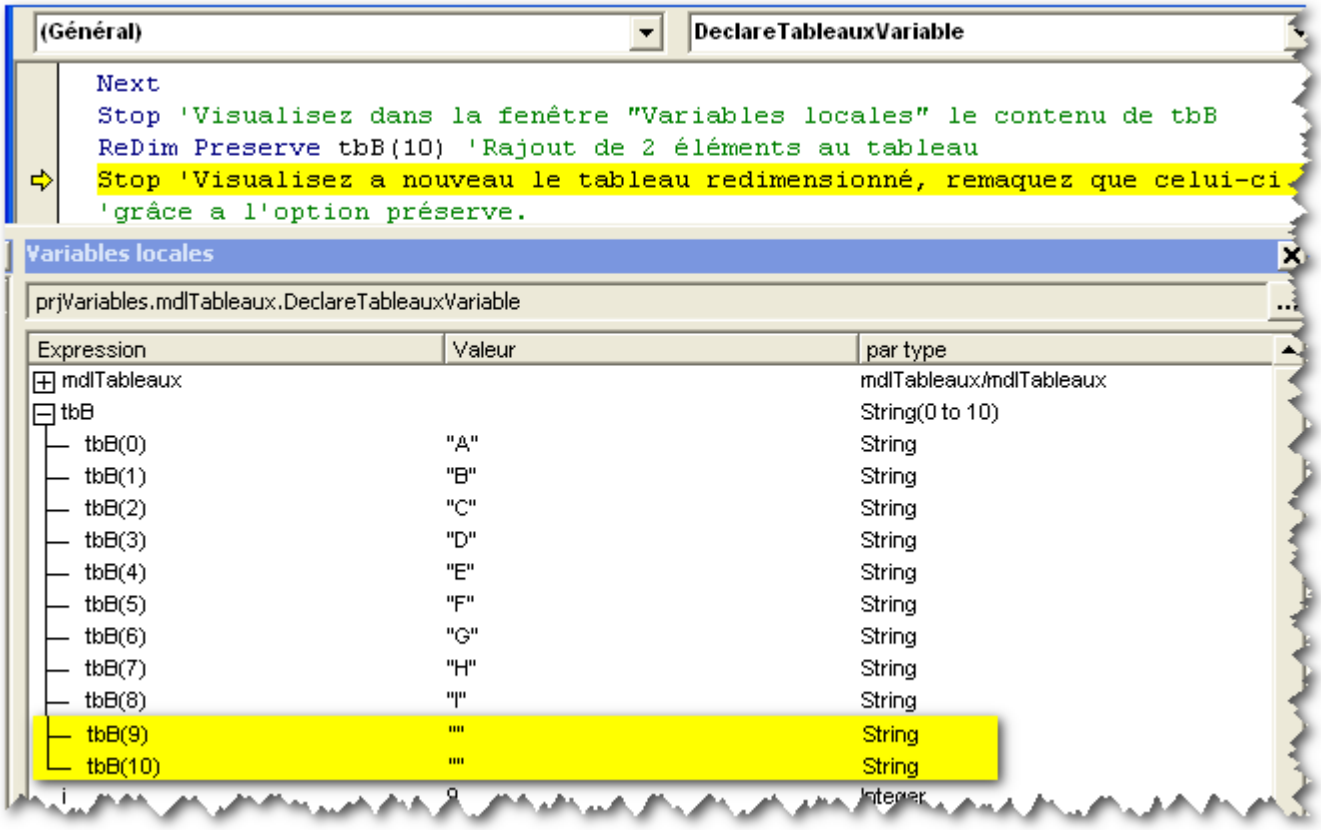
On s'aperçoit, en observant le tableau, que celui-ci a changé de taille (l'index du dernier élément est passé de 6 à 8) cependant, toutes les données contenues précédemment ont été perdues et les divers éléments ont pris la valeur par défaut d'une variable de type string : ""



La suite du code reprend le code d'affectation des divers éléments du tableau.

```
For i = LBound(tbB) To UBound(tbB)
      tbB(i) = Chr(Asc("A") + i)
Next
```

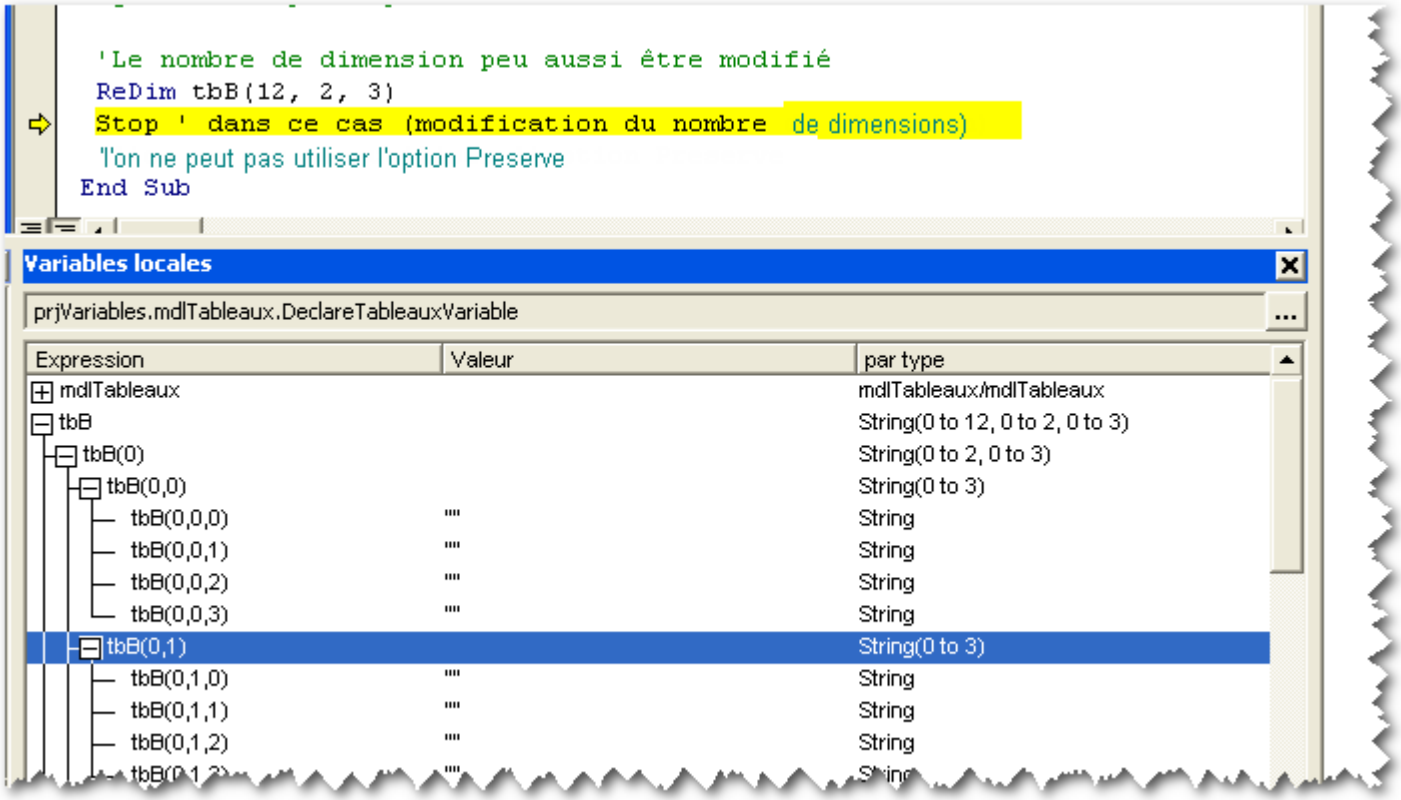
En retour on se retrouve avec un tableau dont les éléments 0 à 8 ont comme valeur les lettres "A" à "I",
la suite démontre l'utilisation de l'option Preserve avec Redim, option qui, comme son nom l'indique permet de préserver le contenu du tableau lors du redimensionnement.



On retrouve bien, suite au passage de 8 à 10 éléments, deux éléments supplémentaires tbB(9) et tbB(10) initialisés avec la valeur "", tandis que les éléments d'index 0 à 8 n'ont pas été modifiés.

Le dernier code permet de modifier le nombre de dimensions du tableau :

```
'Le nombre de dimensions peut aussi être modifié
 ReDim tbB(12, 2, 3)
 Stop ' dans ce cas (modification du nombre de dimensions)
 'l'on ne peut utiliser l'option Preserve
End Sub
```



Sachant que la modification du nombre de dimensions interdit l'utilisation de l'option Preserve.

XV - Télécharger l'application prjVariables

Les sources VB6 du projet prjVariables

ZZ - Conclusion

Z-A - Pour aller plus loin

Cet article ne constitue qu'une base dans laquelle j'espère pouvoir faire vivre le chapitre "VB6 par l'Exemple".
J'ai ouvert une discussion dans laquelle vous pouvez laisser vos commentaires, idées...

Z-B - Remerciements

Je tiens à remercier **Pierre Fauconnier**, **jacques_jean** , **blade159** ainsi que **Caro-Line** pour leur relecture.