Visual Basic pour les Applications

VBA : Visual Basic for Application est le langage de programmation intégré à la suite bureautique Microsoft Office.

Il offre toutes les structures de contrôle d'un langage de Troisième génération et permet de ne pas déclarer les types de variables.

C'est un langage qui manipule des objets propres à chaque type d'application. Avec le tableur, il va falloir être capable de sélectionner un objet « feuille de calcul », une cellule ou un groupe de cellule, voire une ligne ou une colonne et définir la cellule active.

Déclaration des variables et de leur type

Si l'utilisateur ne définit pas explicitement une variables, Visual Basic lui affecte automatiquement le type **Variant**.

Il est toutefois préférable de déclarer les variables pour les raisons suivantes :

- **rapidité des calculs** : La taille du type Variant dépend des données stockées par la variable. Il gère les valeurs allant jusqu'au Double et aux chaînes de 65 535 caractères. Les conversions requises par des données aussi différentes sont effectuées en interne, ce qui est agréable pour le développeur mais prend du temps de calcul est ralentit l'exécution du programme.
- **moindre encombrement de la mémoire**:
- **Compatibilité** : à l'exception de VB, aucun dialecte Basic ne connaît le type **Variant.** Si vous envisagez d'exporter une application VB vers un autre Basic, renoncez à l'usage de **Variant**.

Les types de variables

Attribuer un type à une variable consiste à lui associer une taille mémoire et surtout une plage de valeurs.

Type de donnée	Taille d'enregistrement	Plage
Byte	1 octet	0 à255
Boolean	2 octets	Vrai ou Faux.
Integer	2 octets	-32 768 à 32 767.
Long (entier long)	4 octets	-2 147 483 648 à 2 147 483 647
Single (valeur à virgule flottante en simple précision)	4 octets	-3,402823E38 à -1,401298E-45 pour les valeurs négatives; 1,401298E-45 à 3,402823E38 pour les valeurs positives.
Double (valeur à virgule flottante en double précision)	8 octets	-1,79769313486232E308 à -4,94065645841247E-324 pour les valeurs négatives; 4,94065645841247E-324 à 1,79769313486232E308 pour les valeurs positives.
Currency	8 octets	-922 337 203 685 477,5808 à 922 337 203 685 477,5807.
Date	8 octets	1er janvier 100 au 31 décembre 9999.
Object	4 octets	Toute référence à des données de type Objet.
String	1 octet par caractère	0 à environ 2 milliards (environ 65 535 pour Microsoft Windows versions 3.1 et antérieures).
Variant	16 octets + 1 octet pour chaque caractère	N'importe quelle valeur numérique dans la plage d'une valeur de type Double ou n'importe quel texte de caractères.

Déclaration de variables

C'est le mot-clé **Dim** qui permet de déclarer les variables.

Dim Adresse **As String** .....déclare une variable de texte de longueur variable

Dim Adresse **As String *50**.....la chaîne compte toujours 50 caractères

Dim Nombre **As Integer**.....déclare explicitement une variable de type Entier.

Dim AutreVar;Choix **As Boolean**;DateNaissance **As Date**

déclarations multiples sur une même ligne. La variable AutreVar est de type Variant puisque son type n'est pas spécifié.

Attention !!! Dans une liste de variables de même type, il faut répéter le type pour chaque variable.

Exemple :

Dim Lig , Col As Integer

Dim Alig, ACol As Integer

Dim numéro As Integer

Dim Ordre, reste As Integer

Dim Msg, Title, Default As String

Les Opérateurs

Opérateurs arithmétiques

Opérateur Basic	Description	Ordre de priorité
^	élévation à la puissance *2^3 =2*2*2=8*	1
*	multiplication	2
/	division	2
\	division entière *38\5 = 7*	3
Mod	Reste de la division entière *38 Mod 5 = 3*	4
+	addition	5
-	soustraction	5

Opérateurs de comparaison

<	Inférieur à
<=	Inférieur ou égal à
>	Supérieur à
>=	Supérieur ou égal à
=	Egal à
<>	Différent de

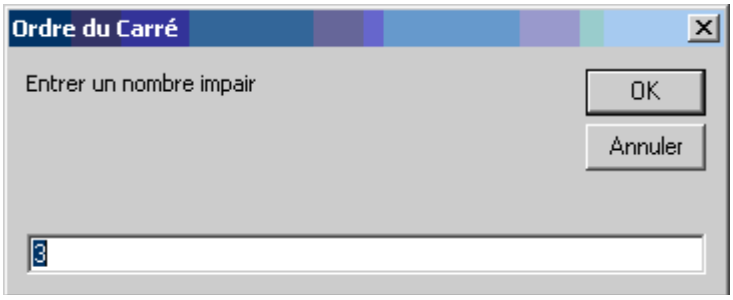
Opérateurs de concaténation

&	Concaténation de chaînes

Opérateurs logiques

And	Conjonction
Eqv	Equivalence
Imp	Implication
Not	Négation
Or	Disjonction (Ou inclusif)
Xor	Exclusion (Ou exclusif)

Introduire une valeur dans une boite de dialogue



Titre de la boite

Message destiné à l'utilisateur

Valeur par défaut

Pour afficher cette boite de dialogue, il faut écrire les instructions suivantes :

prompt = "Entrer un nombre impair" ' *Définit le message.*

Title = "Ordre du Carré" ' *Définit le titre de la boite.*

Default = "3" ‘ *Définit la valeur par défaut.*

xpos = 4000

ypos = 4000

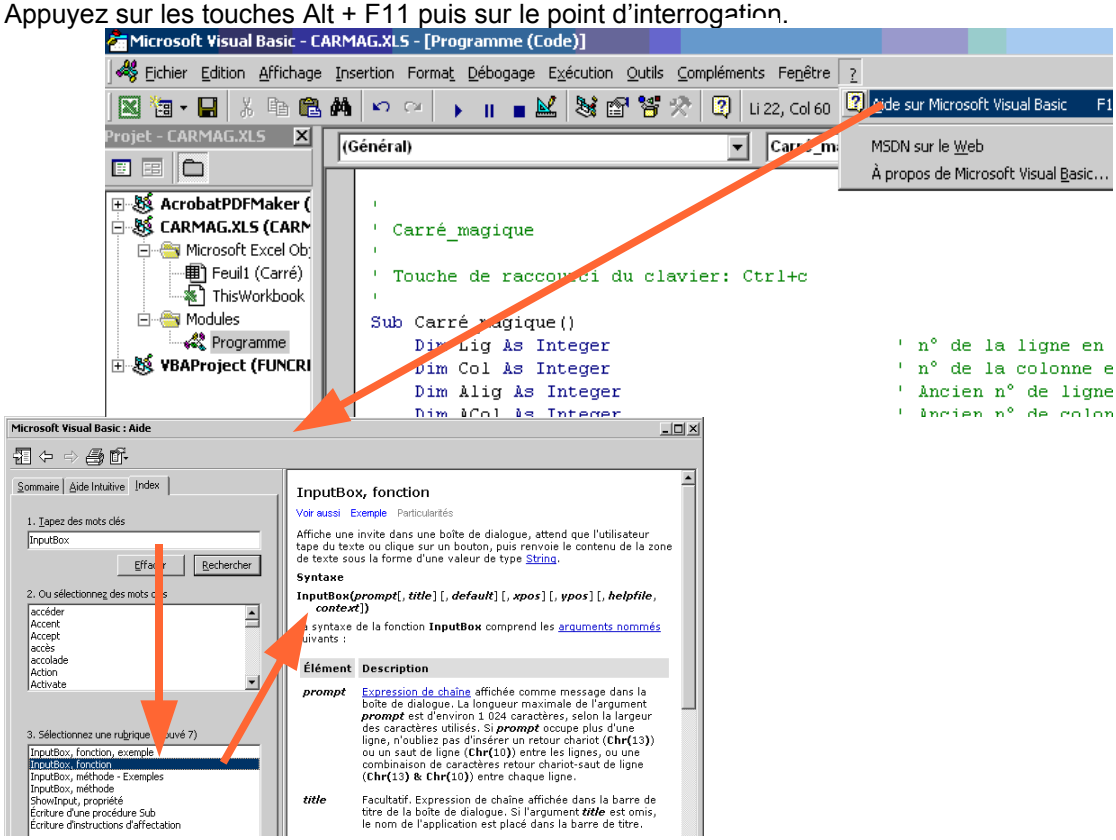
Ordre = InputBox(prompt, Title, Default,xpos,ypos) ' *Affiche le message, le titre et la valeur par défaut.*

On pourrait également tout concentrer en une seule instruction, mais ce serait au détriment de la lisibilité.

Ordre = InputBox("Entrer un nombre impair", "Ordre du Carré", "3",4000,4000)

Pour connaître la syntaxe des instructions ou fonctions de Visual Basic, il faut appeler l'aide à partir de l'éditeur de Code.

Appuyez sur les touches Alt + F11 puis sur le point d'interrogation.



Répétitive REPETER…JUSQU'A

prompt = "Entrer un nombre impair".

Dans le message affiché précédemment, il faut vérifier que l'utilisateur entre bien un nombre impair. Si tel n'est pas le cas, il faut afficher à nouveau la boite de dialogue.

L'affichage du message doit être répétitif tant que le nombre introduit n'est pas impair.

Tel que nous venons de le formuler, une boucle « tant Que » semble s'imposer ! Mais en recherchant les cardinalités d'affichage du message, on s'aperçoit qu'il doit au moins être affiché une fois et réaffiché plusieurs fois selon le comportement de l'utilisateur. **La cardinalité est 1,n**.

La boucle Répéter… Jusqu'à s'impose donc.

Le raisonnement devient :

Répéter

Afficher la boite de dialogue

Jusqu'à ce que le nombre frappé par l'utilisateur soit impair

Syntaxe	Exemple
Do [instruction] [**Exit Do**] [instruction] **Loop** [**Until** condition]	Do Ordre = InputBox(prompt, Title, Default, xpos, ypos) reste = Ordre Mod 2 **Loop Until** reste = 1

Un nombre impair est un nombre dont le reste de la division entière par 2 est 1. L'opérateur « division entière » dite aussi « division Euclidienne » est MOD, abréviation de MODULO.

Si l'utilisateur frappe le nombre 6, la variable Ordre contiendra la valeur 6.

Ordre → 6 2
Reste → 0 3

Reste vaudra 0. Le programme remontera jusqu'à l'ordre Do et réaffichera la boite de dialogue.

Dans le calcul d'un carré magique, il faut après avoir introduit l'ordre qui nous donne la taille du carré, remplir toute les cellules de 0

Nous allons donc balayer toutes les lignes et toutes les colonnes concernées et affecter un 0 à chaque cellule.

Si l'utilisateur frappe la valeur 5, l'Ordre du carré sera de 5, c'est à dire que ce carré possèdera Ordre * Ordre cellules. Il faudra balayer ordre colonnes sur ordre lignes.

La cardinalité, c'est à dire le nombre d'itérations (ou d'opérations) est connu à l'avance. La boucle POUR s'impose !

Répétitive POUR

Syntaxe	Exemple
For MonIndice = 15 **to** 75 **Step** 5 Instructions [**Exit For**] Instructions **Next** MonIndice	**For** Lig = 1 To Ordre **For** Col = 1 To Ordre Cells(Lig, Col).Select ActiveCell.FormulaR1C1 = 0 **Next** Col **Next** Lig
Step est le pas, on dit l'incrément. La première valeur de MonIndice sera 15, la suivante 15 + 5, soit20, puis 25, 30…jusqu'à 75. Si la valeur Step n'est pas précisée , elle est implicitement égale à 1. Exit For , permet de sortir de la boucle avant la dernière valeur atteinte. Exit for est généralement mentionnée dans une condition.	La boucle extérieur, joue sur les numéros de ligne, alors que la boucle interne, passe en revue toutes les cellules d'une ligne. La ligne **Cells(Lig, Col).Select**, rend active la cellule de coordonnée lig, col. La ligne **ActiveCell.FormulaR1C1 = 0**, affecte la valeur 0 à la cellule active.

Après avoir initialisé le carré, il faut écrire la valeur 1 dans la case médiane de la première ligne.

```
Lig = 1                                  ' position de la ligne
Col = Fix(Ordre / 2) + 1                  ' position de la colonne
numero = 1                               ‘ numero de la case
Cells(Lig, Col).Select                    ' position du pointeur de cellule
 ActiveCell.FormulaR1C1 = numero          ' numerotation de la cellule
```

Col = Fix(Ordre / 2) + 1, la fonction Fix correspond à la valeur entière. Ainsi si Ordre vaut 3, Col prendra la valeur partie entière (3 / 2) soit partie entière (1,5) donne 1. Donc 1+1 donne 2. C'est bien la case médiane de la première ligne d'un carré d'ordre 3.

Ce premier calcul correspond à l'amorce de la boucle suivante qui peut éventuellement ne pas être exécutée si l'utilisateur demande un carré d'ordre 1. La cardinalité de la boucle est **0, N.**

La boucle Tant Que s'impose. On l'utilisera tant que le numéro de la cellule à numéroter est inférieure au carré de l'ordre.

Tant que numero < ordre^2

    Numéroter la case suivante
Fin tant Que

Répétitive TANT…QUE

```
Compteur = 0                          ' Initialise la variable.
Do While Compteur < 20                ' Vérifie la valeur de Compteur.
    Compteur = Compteur + 1           ' Incrémente Compteur.

Loop                                  ‘ plancher de boucle
```

Exemple

```
Vérifier = Vrai
Compteur = 0                                    ' Initialise les variables.
Do.                                             ' Boucle externe.
Do While Compteur < 20                          ‘ Boucle interne.
Compteur = Compteur + 1                         ' Incrémente le compteur.
If Compteur = 10 Then                           ' Si la condition est vraie.
Vérifier = Faux.                                ' Affecte la valeur Faux à l'indicateur.
Exit Do                                         ' Sort de la boucle interne.
End If
Loop                                            ‘ plancher de la boucle TantQue
Loop Until Vérifier = Faux                      ' Sort immédiatement de la boucle externe.
```

Dans cet exemple, l'instruction Do While...Loop incrémente une variable de compteur.

Les instructions incluses dans la boucle sont exécutées tant que la condition est vraie.

Pour la numérotation des cellules du carré magique, trois cas d'exception peuvent se présenter :

1. Le numéro de la ligne est égal à 0, il faut alors lui donner la valeur de l'ordre ;
2. Le numéro de colonne est égal à ordre + 1, il prendra alors la valeur zéro ;
3. La cellule à numéroter l'est déjà, il faut donc revenir à la cellule précédente.

Ces trois situations peuvent être gérées par un raisonnement alternatif : **Si Alors Sinon.**

Si....Alors...Sinon....Fin Si

VBA supporte les 4 syntaxes suivantes :

Syntaxe 1 : une seule ligne
 	
If condition **Then**, InstructionsVrai
	Syntaxe 2 :
	If condition **Then**
	Instructions
	End If

Syntaxe3
 	
If condition **Then**
Instructions
Else
Instructions
End If
	Syntaxe 4 :
	
	If condition1 **Then**
	Instructions1
	Elseif condition2 **Then**
	Instructions2
	Else
	InstructionsAutres
	End If

```
Lig = Lig - 1
Col = Col + 1
numero = numero + 1
```

Ces deux lignes calculent le déplacement en diagonale. Le pointeur remonte d'une ligne et se déplace d'une colonne.

On calcule ainsi le numéro de la prochaine case.

```
If Lig = 0 Then
        Lig = Ordre
End If
```

Le déplacement précédent peut avoir généré un indice de ligne = 0.

On est donc sorti du carré « par le haut ». Il faut alors donner au numéro de ligne la valeur de l'ordre du carré

```
If Col = Ordre + 1 Then
        Col = 1
End If
```

Le déplacement précédent peut avoir généré un indice de colonne égal à ordre + 1. On est sorti par la droite du carré. Il faut revenir en colonne 1.

Le programme complet du carré magique devient :

Les lignes oranges correspondent au programme, les lignes vertes aux commentaires acceptés par l'interpréteur, les lignes bleus aux commentaires de l'auteur.

Algorithme	Programme VBA

Variable carre(21, 21) : numérique	' Carré_magique
	'
	' Touche de raccourci du clavier: Ctrl+c
	'
	Sub Carré_magique()
	  Dim Lig As Integer
	  Dim Col As Integer
	  Dim Alig As Integer
	  Dim ACol As Integer
	  Dim numero As Integer
	  Dim reste, Ordre, xpos, ypos As Integer
	  Dim prompt, Title, Default As String

'---------- ' Introduire l'ordre du Carré et vérifier l'imparité '----------	prompt = "Entrer un nombre impair" Title = "Ordre du Carré". Default = "3" xpos = 4000 ypos = 4000

Répéter
  Introduire ordre
Jusqu'à ce que reste(ordre/2) =1
	Do
	  Ordre = InputBox(prompt, Title, Default, xpos, ypos)
	  reste = Ordre Mod 2
	Loop Until reste = 1

'----------
' Initialisation du carré
'----------
	Sheets("Carré").Select
	  Sheets("Carré").Cells.Select
	  Selection.ClearContents
	' Initialisation du carré

Pour ligne de 1 à ordre
  Pour colonne de 1 à ordre
  carre(ligne, colonne) <- 0
  Fin Pour
Fin Pour
	For Lig = 1 To Ordre
	  For Col = 1 To Ordre
	  Cells(Lig, Col).Select
	  ActiveCell.FormulaR1C1 = 0
	  Next Col
	Next Lig
	Application.Wait (Now + TimeValue("0:00:01"))

'---------- ' Numérotation de la case 1 '----------	Lig = 1 Col = Fix(Ordre / 2) + 1 numero = 1 Cells(Lig, Col).Select Debug.Print "i-lig = "; Lig, "col ="; Col ActiveCell.FormulaR1C1 = numero
ligne <- 1 colonne <- Entier(ordre / 2) + 1 numéro_case <- 1 carre(ligne, colonne) <- numéro_case	
'---------- ' Numérotation des autres cases jusqu'à ordre*ordre '----------	
Tant que numéro_case <= ordre * ordre	Do While numero < Ordre * Ordre
'---------- ' Mémorisation des anciennes coordonnées '----------	
anc_ligne <- ligne anc_colonne <- colonne	Alig = Lig ACol = Col
'---------- ' Déplacement en diagonale '----------	
ligne <- ligne - 1 colonne <- colonne + 1 numéro_case <- numéro_case + 1	Lig = Lig - 1 Col = Col + 1 Debug.Print "0-lig = "; Lig, "col ="; Col numero = numero + 1
'---------- ' Si sortie par le haut '----------	

Si ligne = 0 Alors
  ligne <- ordre
Fin Si
	If Lig = 0 Then
	  Lig = Ordre
	  Debug.Print "1-lig = "; Lig, "col ="; Col
	End If

'---------- ' Si sortie à droite '----------	

Si colonne = ordre + 1 Alors
  colonne <- 1
Fin Si
	If Col = Ordre + 1 Then
	  Col = 1
	  Debug.Print "2-lig = "; Lig, "col ="; Col
	End If

'---------- ' Si après déplacement la case est déjà numérotée on ' se place sous la dernière case numérotée '----------	

```
Si carre(ligne, colonne) <> 0 Alors
        ligne <- anc_ligne + 1
        colonne <- anc_colonne
    Fin Si

    carre(ligne, colonne) <- numéro_case


Fin Tant Que
  '-----------------------------------------------------
  '  Affichage du carré magique
  '-----------------------------------------------------
Pour  ligne de 1 à ordre
    Pour colonne de 1 à ordre
            Ecrire(carre(ligne, colonne )
    Fin Pour
Fin Pour
  '-----------------------------------------------------
  'Calcul de la valeur magique
  '-----------------------------------------------------
  magie <– 0
  Pour compteur de 1 à ordre
    magie <– magie + carre(1, compteur)
  Fin Pour
  Ecrire " Valeur magique = ", magie
Fin
```

```
    If Cells(Lig, Col) <> 0 Then
      Lig = Alig + 1
      Col = ACol
      Debug.Print "3-lig = "; Lig, "col ="; Col
    End If

    Cells(Lig, Col).Select
    Debug.Print "4-lig = "; Lig, "col ="; Col
    ActiveCell.FormulaR1C1 = numero
  Loop
```

L'affichage du carré n'est pas nécessaire dans Excel.
L'instruction *ActiveCell.FormulaR1C1 = numero, qui* affecte une valeur à la cellule active implique automatiquement l'affichage de cette valeur.
C'est une particularité du Tableur.

```
End Sub
```

Lignes de commentaire

Le caractère apostrophe, indique à l'interpréteur du langage, un début de commentaire. Tout caractère compris entre une apostrophe et un retour à la ligne sera considéré comme du texte libre.

Vous pouvez dès lors consacrer des lignes entières aux commentaires, ou insérer un commentaire en fin de ligne.

Dans l'Editeur VBA, les commentaires apparaissent en vert

```
' Touche de raccourci du clavier: Ctrl+c
'

Sub Carré_magique()
   Dim Lig As Integer                ' n° de la ligne en cours
   Dim Col As Integer                ' n° de la colonne en cours
```

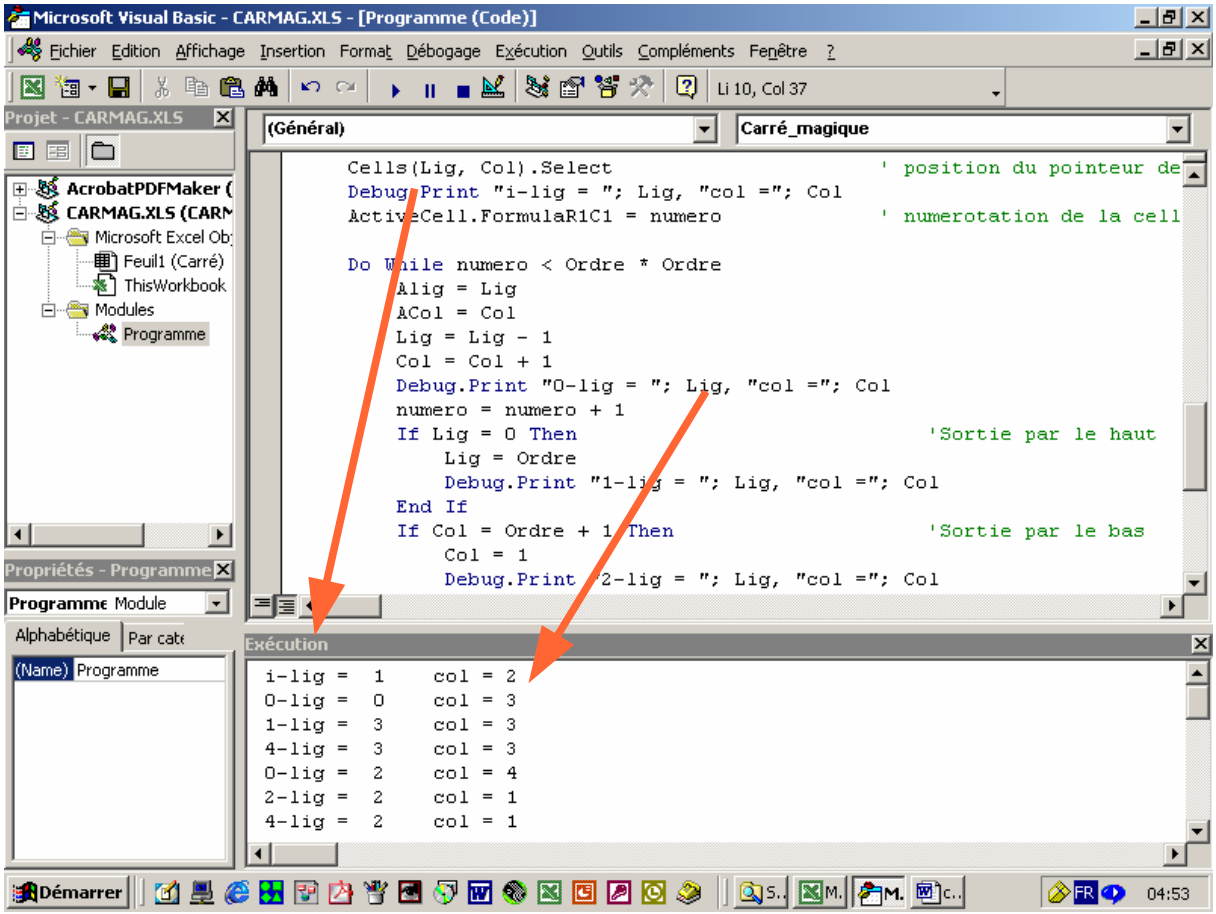
Visualisation des valeurs des variables

Pour vérifier le bon fonctionnement de son programme et vérifier la valeur prise par les variables calculées, il est possible d'ajouter une instruction Debug.print, dont le but est d'afficher des valeurs dans la fenêtre d'exécution du programme.

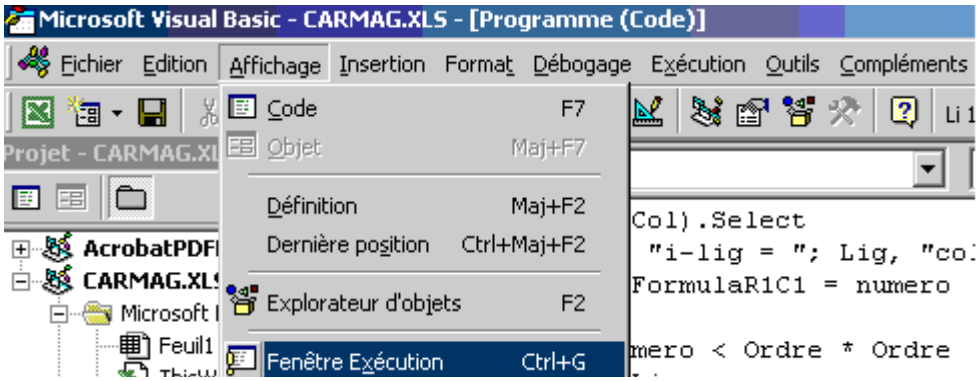
```
Debug.Print "3-lig = "; Lig, "col ="; Col
```

Le texte entre guillemets est reproduit tel quel, alors que les variables sont remplacées par la valeur calculée.

Debug.Print	" **3-lig =**"	;	Lig	"**col=**"	;	Col
Imprimer dans la fenêtre d'exécution	Reproduire tel quel	tabulation	Valeur de la variable Lig	Reproduire tel quel	tabulation	Valeur de la variable Col



La fenêtre d'exécution est accessible à partir du menu de commande **Affichage.**



Temporisation d'exécution

Application.Wait (Now + TimeValue("0:00:01")) est une instruction qui permet de générer un e attente dans l'exécution d'un programme.

Dans l'exemple précédent le temps d'attente est de une seconde.

Nous l'utilisons dans le carré magique pour permettre à l'utilisateur de voir le résultat de la phase d'initialisation.