aprenderaprogramar.com

Editor de código Visual Basic. Lista de miembros automática, sugerencias de sintaxis, autocompletado y otras opciones. (CU00313A)

Sección: Cursos

Categoría: Curso Visual Basic Nivel I

Fecha revisión: 2029

Autor: Mario R. Rancel

Resumen: Entrega nº12 del Curso Visual Basic Nivel I

29

ASISTENCIA DE VISUAL BASIC PARA ESCRITURA DE CÓDIGO

Ya hemos visto dónde debemos escribir el código asociado a un formulario. Hay que destacar que Visual Basic facilita el que podamos escribir código con rapidez a través de:

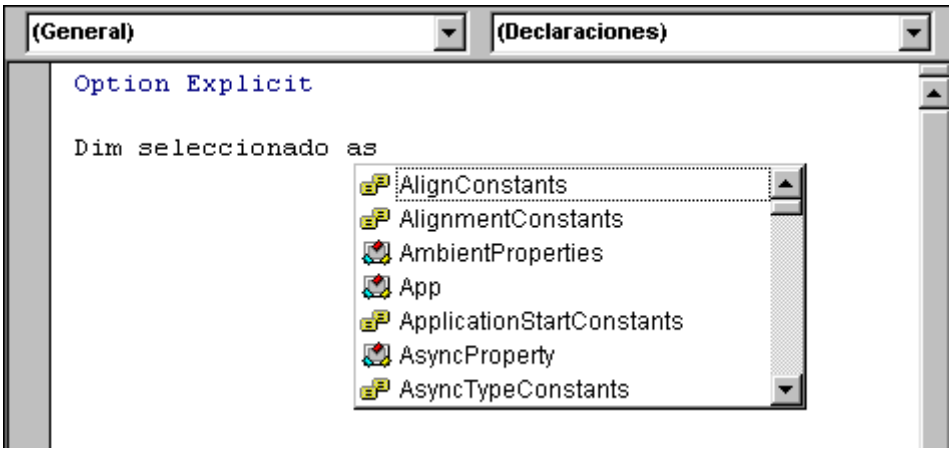
- Propuestas para elección de tipos de variable, propiedades de un objeto, etc.
- Corrección o indicación automática de errores que podamos cometer.
- Adelanto de la terminación del nombre de instrucciones o variables que hemos empezado a escribir.

Para que esta asistencia en la escritura esté disponible deberás comprobar:

En las versiones menos recientes, en el menú Herramientas - Opciones..., ventana Editor - Opciones del código), que tienes activadas las opciones correspondientes, siendo las principales: Comprobación automática de sintaxis (Habilitar sugerencias de corrección de errores), Lista de miembros automática, Información rápida automática y Sugerencias de datos automáticas.

En las versiones más recientes en el menú Herramientas, Opciones, activa la casilla “Mostrar todas las configuraciones”. A continuación elige Editor de texto, Basic, y comprueba que tienes activadas las opciones correspondientes, siendo las principales: General-Lista de miembros automática, General-Información de parámetros, Opciones Específicas – Inserción automática de construcciones End, Lista descriptiva (nuevo formato) de código, Habilitar sugerencias de corrección de errores y Habilitar resaltado de referencias y palabras clave.

Una vez verificado esto, vamos a comprobar cómo nos ayuda Visual Basic a escribir código. Dirígete a la ventana de código y escribe: Dim seleccionado As. Comprobarás cómo aparece una ventana donde existen diferentes posibilidades:



A través del cursor o del ratón, selecciona Boolean, con lo cual verás que se escribe automáticamente Boolean sin necesidad de teclearlo. Por tanto Visual Basic ha interpretado que ibas a definir el tipo de dato para la variable seleccionado y te ha mostrado las distintas posibilidades que tenías, facilitando la elección. Borra ahora la declaración de la variable y escribe lo siguiente:

```
Dim seleccionado as boolean
```

Pulsa enter y comprobarás que el texto automáticamente se convierte en:

```
Dim seleccionado As Boolean
```

Es decir, después de haber cometido un error al no poner correctamente las mayúsculas en as boolean, Visual Basic ha corregido automáticamente la escritura adaptándola a la sintaxis correcta. Esto es muy útil, ya que nos basta con tener las ideas claras respecto a lo que queremos escribir y saber su sintaxis aproximada. De los "detalles de sintaxis" se encargará Visual Basic, permitiendo que centremos nuestra atención en los aspectos principales de la programación.

Las correcciones no son (no pueden serlo) siempre automáticas. En algunos casos nos aparecerá un mensaje de error indicativo de que Visual Basic no entiende esa sintaxis y en otros no detectaremos que hemos cometido un error hasta tratar de ejecutar el programa.

Escribe por ejemplo: Dik seleccionado As Boolean

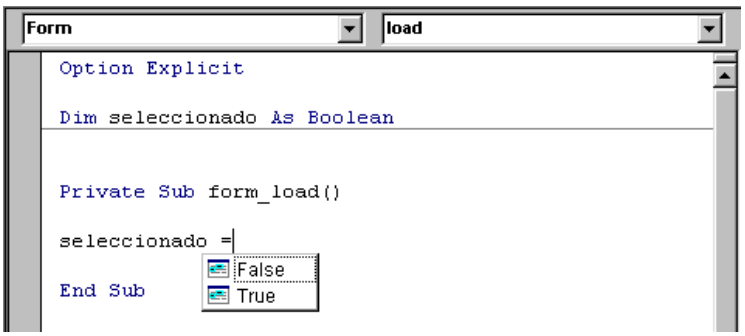
No habrá corrección automática pero al pulsar enter te aparecerá el texto remarcado o subrayado indicando que Visual Basic detecta que “hay algo extraño” en el código. Esto te servirá para prestar atención y darte cuenta de que en lugar de Dim has escrito Dik.

Si escribes Dim seleccionado As Voolean no hay corrección automática. Quizás aparezca un subrayado o remarcado indicando que se detecta algo extraño (o quizás no), ya que Visual Basic no sabe si tú has creado un tipo de variable al que has llamado Voolean. El error en este caso te saltará, si no lo corriges antes, al tratar de ejecutar el programa.

El editor de Visual Basic también nos puede ayudar a completar fragmentos de código obligatorios. Por ejemplo, en las versiones menos recientes cuando escribimos Private Sub form_load() y pulsamos enter nos aparece automáticamente una nueva línea End Sub, que debe acompañar obligatoriamente a la línea Private Sub form_load(). En las versiones más recientes si escribimos if seleccionado = True y pulsamos enter automáticamente nos aparecerá If seleccionado = True Then y en otra línea End If. Ya que son fragmentos de código que obligatoriamente tenemos que escribir, Visual Basic nos ahorra este trabajo. Aunque si lo preferimos, podemos desactivar ese autocompletado automático.

Si definimos seleccionado como una variable tipo Boolean y escribimos en el editor: seleccionado =

Comprobarás que aparece una ventanita con dos opciones: false y true.



Esto es así por estar la variable definida como tipo Boolean. Son los dos valores que se le podrían asignar y Visual Basic nos lo recuerda. Cuando estamos trabajando con decenas de variables es muy

útil. Podemos seleccionar False o True haciendo uso del cursor o del ratón, sin necesidad de escribirlo. Estas sugerencias automáticas podemos activarlas o desactivarlas según deseemos.

ORDENACIÓN DE LÍNEAS. NUMERACIÓN DE LÍNEAS. ETIQUETAS DE LÍNEAS.

Caso 1 versiones menos recientes:

```
Rem Curso Visual Basic aprenderaprogramar.com
Option Explicit
Dim seleccionado As Boolean

Private Sub Form_Load()
seleccionado = True
MsgBox ("El valor actual de seleccionado es " & seleccionado)
End Sub
```

Caso 1 versiones más recientes:

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1
    Dim seleccionado As Boolean

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
        seleccionado = True
        MsgBox("El valor actual de seleccionado es " & seleccionado)
    End Sub
End Class
```

Caso 2 versiones menos recientes:

```
Rem Curso Visual Basic aprenderaprogramar.com
Option Explicit
Dim seleccionado As Boolean

Private Sub Form_Load()
10 seleccionado = True
20 MsgBox ("El valor actual de seleccionado es " & seleccionado)
End Sub
```

Caso 2 versiones más recientes:

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1
    Dim seleccionado As Boolean

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
10:     seleccionado = True
20:     MsgBox("El valor actual de seleccionado es " & seleccionado)
    End Sub
End Class
```

Caso 3 versiones menos recientes:

```
Rem Curso Visual Basic aprenderaprogramar.com
Option Explicit
Dim seleccionado As Boolean

Private Sub Form_Load()
Líneaasignacontenido: seleccionado = True
Líneamuestramensaje: MsgBox ("El valor actual de seleccionado es " & seleccionado)
End Sub
```

Caso 3 versiones más recientes:

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1
    Dim seleccionado As Boolean

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
Líneaasignacontenido: seleccionado = True
Líneamuestramensaje: MsgBox("El valor actual de seleccionado es " & seleccionado)
    End Sub
End Class
```

Caso 4 versiones menos recientes:

```
Rem Curso Visual Basic aprenderaprogramar.com
Option Explicit
Dim seleccionado As Boolean

Private Sub Form_Load()
10 seleccionado = True
Líneamuestramensaje: MsgBox ("El valor actual de
seleccionado es " & seleccionado)
End Sub
```

Caso 4 versiones más recientes:

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1
    Dim seleccionado As Boolean

    Private Sub Form1_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
10:     seleccionado = True
Líneamuestramensaje: MsgBox("El valor actual
de seleccionado es " & seleccionado)
    End Sub
End Class
```

Los cuatro casos presentados son distintas posibilidades para numerar o etiquetar las líneas. Escribe el código en tu editor y compruébalo tú mismo.

El caso 1 corresponde a un programa en el que no existe identificación de las líneas, que para nosotros será lo habitual (en general es infrecuente identificar las líneas).

El caso 2 corresponde a un programa con las líneas numeradas (en las versiones más recientes después del número de línea se incluyen dos puntos). Recuerda a lo que era el Basic tradicional, para el que la numeración de líneas tenía una importancia notable al ser un tipo de programación altamente dependiente de la instrucción Go To (IrA), circunstancia que ya no se da con Visual Basic. Por tanto, en general no usaremos este tipo de numeración aunque Visual Basic la permite.

El caso 3 corresponde a un programa con las líneas identificadas por etiquetas. Cada línea tiene un nombre descriptivo que le precede, separado del contenido del código en sí por un signo de dos puntos. En circunstancias puntuales puede ser de interés, pero tampoco tiene sentido usarlo de forma constante para programar.

El caso 4 corresponde a un programa donde algunas líneas están numeradas, otras identificadas por una etiqueta y otras carecen de identificación.

Las posibilidades que se nos abren son amplias. Nosotros prescindiremos de la identificación de líneas, excepto para cuestiones puntuales como podría ser direccionar la gestión de errores. Sobre gestión de errores hablaremos más adelante.

Próxima entrega: CU00314A

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61