Apuntes Programación Excel VBA

PARTE VI: Código de interés (I). Generalidades

Jose Ignacio González Gómez
Departamento de Economía Financiera y Contabilidad - Universidad de La Laguna
www.jggomez.eu

INDICE

1 Código VBA, casos y ejemplos simples

Presentamos una selección de **ejemplos de código en VBA** listo para ejecutar mediante cualquier macro o por medio de un evento de algún objeto (Excel, un libro, una hoja o un rango).

En este artículo, ofrecemos líneas de código ejecutable para realizar acciones con una hoja de Excel, siempre que se cumpla una condición.

El texto en verde que a veces se puede ver entre el código, es el que debe personalizar el usuario, si procede.

1.1 Condiciones con IF...THEN...ELSE

1.1.1 Comprobar que un número escrito en la celda activa es el requerido

Comprobar que un número escrito en la celda activa es el requerido. Si se cumple, muestra un mensaje diciendo "El código es correcto". Si no se cumple termina y no hace nada.

```
Sub ComprobarCódigo()
  If ActiveCell = "BF1234" Then
    MsgBox ("El código es correcto")
  End If
End Sub
```

1.1.2 Comprobar que un número escrito en la celda activa es el requerido en caso contrario mostrar error

Comprobar que un número escrito en una celda determinada es el requerido. Si se cumple, muestra un mensaje diciendo "El código es correcto". Si no se cumple, muestra un mensaje diciendo "El código no es correcto".

```
Sub ComprobarCódigo()
  If Range("A1") = "BF1234" Then
    MsgBox("El código es correcto")
  Else
    MsgBox("El código no es correcto")
  End If
End Sub
```

1.2 Acciones con rangos de Excel, código VBA

Veremos formas de SELECCIONAR, BORRAR y ESCRIBIR EN CELDAS.

1.2.1 Seleccionar rangos en Excel con VBA. Una celda, una columna o fila, rangos distintos, etc.

1) Seleccionar una celda

```
Sub SeleccionarCeldaA1()
  Range("A1").Select
End Sub
```

2) Seleccionar una celda que está 2 filas más arriba y una columna más a la derecha que la celda actualmente seleccionada.

```
Sub CambiarSeleccionCelda()
  ActiveCell.Offset(-2, 1).Select
End Sub
```

3) Seleccionar toda la columna A.

```
Sub SeleccionarColumnaA()
  Range("A:A").Select
End Sub
```

4) Seleccionar toda la fila 1.

```
Sub SeleccionarFila1()
  Range("1:1").Select
End Sub
```

5) Seleccionar rangos distintos.

```
Sub SeleccionarRangosDistintos()
  Range("A1:A4,B5,C4:D4").Select
End Sub
```

1.2.2 *Edición de Rangos en Excel con VBA. Borrar e introducir contenido, etc*

6) Borrar contenido de un rango.

```
Sub BorrarContenidoRango()
  Selection.ClearContents
End Sub
```

7) Borrar fila seleccionada.

```
Sub BorrarFila()
  Selection.EntireRow.Delete
End Sub
```

8) Borrar columna seleccionada.

```
Sub BorrarFila()
  Selection.EntireColumn.Delete
End Sub
```

9) Introducir valor en celda 1

Introducir el valor 1000 en la celda A1 de la hoja activa (si lo que se introduce es texto, debe ir entre comillas y si lo que se introduce es un número seguido de un texto, deben ir separados por el símbolo &).

```
Sub IntroducirValor()
  Range("A1") = 1000
End Sub
```

10) Introducir valor en celda 2

Introducir el valor 1000 en la celda A1 de la hoja2 (si lo que se introduce es texto, debe ir entre comillas y si lo que se introduce es un número seguido de un texto, deben ir separados por el símbolo &).

```
Sub IntroducirValor()
  Sheets("Hoja2").Range("A1") = 1000
End Sub
```

1.3 Acciones con una hoja de Excel, código VBA

En este apartado presentamos como realizar acciones con una hoja de como OCULTAR, MOSTRAR, PROTEGER, DESPROTEGER, SELECCIONAR, IMPRIMIR etc.

1.3.1 Ocultar/Mostrar hojas

Ocultar una hoja de Excel (para mostrarla, hay que cambiar "False" por "True".

```
Sub OcultarHoja1()
  Sheets("Hoja1").Visible=False
End Sub
```

Ocultar totalmente una hoja de Excel (no se podrá mostrar con el menú contextual que emerge de las pestañas de hojas).

```
Sub OcultarHoja1Totalmente()
  Sheets("Hoja1").Visible=xlVeryHidden
End Sub
```

1.3.2 Protección de hojas

Proteger la hoja activa de Excel (no se podrán seleccionar las celdas bloqueadas).

```
Sub ProtegerHoja()
  ActiveSheet.Protect  DrawingObjects:=True,  Contents:=True,  Scenarios:=True
  ActiveSheet.EnableSelection = xlUnlockedCells
End Sub
```

1.3.3 Desproteger una hoja de Excel.

```
Sub DesprotegerHoja()
  ActiveSheet.Unprotect
End Sub
```

1.3.4 Varias acciones con las hojas, Seleccionar una hoja, cambiar el zoom

Seleccionar una hoja de Excel (el nombre debe ser el que aparece en la pestaña de la hoja).

```
Sub SeleccionarHoja1()
  Sheets("Hoja1").Select
End Sub
```

Cambiar el zoom de una hoja de Excel

```
Sub Zoom120()
  ActiveWindow.Zoom = 120
End Sub
```

1.3.5 Imprimir la hoja seleccionada, ocultar la cuadricula,

Imprimir las hojas seleccionadas con la impresora y configuraciones por defecto.

```
Sub ImprimirHojaActiva()
  ActiveWindow.SelectedSheets.PrintOut copies:=1, collate:=True
End Sub
```

1.3.6 Ocultar la cuadrícula de la hoja activa

Ocultar la cuadrícula de la hoja activa (para mostrarlos, cambiar "False" por "True").

```
Sub OcultarCuadricula()
  ActiveWindow.DisplayGridlines = False
End Sub
```

1.3.7 Ocultar encabezamiento de filas y columnas

Ocultar los títulos de encabezamiento de filas y columnas (para mostrarlos, cambiar "False" por "True").

```
Sub OcultarTitulos()
  ActiveWindow.DisplayHeadings = False
End Sub
```

1.4 Acciones con un libro, código VBA

En este apartado, ofrecemos líneas de código ejecutable para realizar acciones con un libro o archivo de Excel como por ejemplo CERRAR, ABRIR, GUARDAR etc.

1.4.1 Cerrar un libro

Cierra un libro de Excel llamado "Libro1.xlsx" (comprueba primero que está abierto). Si ha habido cambios en dicho libro, pregunta si se desea guardarlos.

```
Sub CerrarLibro1()
  Dim TestWorkbook As Workbook
   Set TestWorkbook = Nothing
   On Error Resume Next
   Set TestWorkbook = Workbooks("Libro1.xlsx")
   On Error GoTo 0
   If TestWorkbook Is Nothing Then
     MsgBox "El archivo no estaba abierto"
   Else
     Workbooks("Libro1.xlsx").Close
   End If
End Sub
```

1.4.2 Cierra el libro que esta activo en ese momento

Cierra el libro de Excel que esta activo en este momento (pregunta si guarda los cambios).

```
Sub CerrarLibroActivo()
  ActiveWorkbook.Close
End sub
```

Cierra el libro de Excel que esta activo en este momento y guarda los cambios.

```
Sub CerrarYGuardar()
  ActiveWorkbook.Close savechanges:=True
End Sub
```

1.4.3 Abrir libro en la misma ruta o carpeta

Abrir un libro de Excel llamado "Libro1.xlsx" que está en la misma carpeta o ruta que el libro en el que estamos trabajando.

```
Sub AbrirLibro1MismaRuta()
  Dim TestWorkbook As Workbook
   Set TestWorkbook = Nothing
   On Error Resume Next
```

```
    Set TestWorkbook = Workbooks("Libro1.xlsm")
    On Error GoTo 0
    If TestWorkbook Is Nothing Then
      ruta = ActiveWorkbook.Path
      Workbooks.Open Filename:=ruta & "\Libro1.xlsx"
    Else
      MsgBox "El archivo ya estaba abierto"
    End If
  End sub
```

1.4.4 *Abrir libro que este en otra ruta o carpeta que el libro activo*

Abre un libro de Excel llamado "Libro1.xlsx" que está en la ruta :

C:\Carpeta1\Carpeta2\Carpeta3

```
  Sub AbrirLibro1RutaConcreta()
    Dim TestWorkbook As Workbook
    Set TestWorkbook = Nothing
    On Error Resume Next
    Set TestWorkbook = Workbooks("Libro1.xlsm")
    On Error GoTo 0
    If TestWorkbook Is Nothing Then
      ruta = "C:\Carpeta1\Carpeta2\Carpeta3"
      Workbooks.Open Filename:=ruta & "\Libro1.xlsx"
    Else
      MsgBox "El archivo ya estaba abierto"
    End If
  End sub
```

1.4.5 *Abrir cualquier libro de Excel a través del cuadro de dialogo "Abrir archivo"*

Abrir cualquier libro de Excel (se abre el cuadro de diálogo "Abrir archivo" para elegirlo).

```
  Sub AbrirArchivo()
    Application.Dialogs(xlDialogOpen).Show
  End Sub
```

1.4.6 *Guardar libro activo*

Guarda los cambios en el libro de Excel activo.

```
  Sub GuardarArchivo()
    ActiveWorkbook.Save
  End sub
```

1.4.7 *Guardar libro a través del cuadro de dialogo "Guardar como"*

Guarda los cambios en el libro de Excel activo y abre el diálogo "Guardar como" para hacer una copia de seguridad en una ubicación a elegir.

```
  Sub GuardarComo()
    ActiveWorkbook.Save
    Application.Dialogs(xlDialogSaveAs).Show
  End sub
```

1.5 *Acciones con la aplicación Excel, código VBA. Modificar la apariencia de Excel y salir de la aplicación.*

En este apartado introducimos al lector a conocer las principales líneas de código para realizar acciones con la aplicación, es decir, con el propio Excel. Concretamente veremos como modificar la APARIENCIA DE EXCEL y como SALIR DE EXCEL con VBA.

1.5.1 *Evitar que la pantalla parpadee*

Evitar que la pantalla parpadee o muestre actividad mientras se ejecuta un procedimiento o macro

```
Sub EvitarParpadeoPantalla()
  Application.ScreenUpdating = False
End sub
```

1.5.2 *Ocultar la barra de estado*

Ocultar la barra de estado para ganar espacio útil de pantalla (cambiar "False" por "True" para mostrarla).

```
Sub OcultarBarraEstado()
  Application.DisplayStatusBar = False
End sub
```

1.5.3 *Ocultar la barra de fórmulas*

Ocultar la barra de fórmulas para ganar espacio útil de pantalla (cambiar "False" por "True" para mostrarla).

```
Sub OcultarBarraFormulas()
Application.DisplayFormulaBar = False
End sub
```

1.5.4 *Pasar a pantalla completa*

Ver en modo pantalla completa (cambiar "True" por "False" para salir del modo Pantalla completa).

```
Sub PantallaCompleta()
  Application.DisplayFullScreen = True
End sub
```

1.5.5 *Maximizar Excel*

```
Sub Maximizar()
  Application.WindowState = xlMaximized
End sub
```

1.5.6 *Ocultar pestañas de las hojas*

Ocultar las pestañas de las hojas (cambiar "False" por "True" para mostrarlas).

```
Sub Maximizar()
  ActiveWindow.DisplayWorkbookTabs = False
End sub
```

1.5.7 Copiar-Pegar

Quitar el modo "Copiar-Pegar" (quitar el recuadro de puntos intermitentes que se queda en una celda cuando la copiamos al portapapeles y la tenemos dispuesta para pegar).

```
Sub QuitarModoCopiarPegar()
  Application.CutCopyMode = False
End sub
```

1.5.8 Salir de Excel con VBA

Salir de Excel (si ha habido cambios, pregunta si los guarda).

```
Sub SalirDeExcel()
  Application.Quit
End sub
```

1.5.9 Salir de Excel guardando los cambios

```
Sub SalirDeExcel()
  ActiveWorkbook.Save
  Application.Quit
End sub
```

2 Cómo automatizar una acción al abrir Excel: ir a una hoja concreta, introducir fecha actual en una celda y pasar a modo pantalla completa

Es muy frecuente que cada vez que entramos a una aplicación de Excel , tengamos que realizar una o varias acciones siempre de la misma forma. Por ejemplo, seleccionar una hoja, poner la fecha del día en un determinado campo, mostrar un cuadro de diálogo, etc.

En este artículo, vamos a ofrecer unos primeros pasos muy sencillos en programación Visual Basic for Appliations (VBA) en Excel, para conseguir que al abrir Excel, se ejecute la acción deseada.

Pongamos un sencillo ejemplo, en el que deseamos que al abrir Excel se realicen las siguientes acciones:

1. Seleccionar la Hoja3.
2. Introducir la fecha del día en la celda A1 de la Hoja3.
3. Activar el modo Pantalla completa de Excel.

(Además, deseamos que el usuario no perciba en pantalla que se realiza ninguna de las dos primeras acciones)

Es posible conseguir esto de dos formas. En este artículo lo vamos a hacer creando un procedimiento para el evento "Open" de un libro (también se puede hacer creando una macro y asignándole el nombre de "Auto_open", aunque esta opción puede tener alguna incompatibilidad en algunas circunstancias: Limitaciones de una macro Auto_open.)

En resumen, el proceso es decirle a Excel mediante código VBA, que al abrir el libro (Workbook), ejecute las instrucciones que nosotros queramos. Esto se hace introduciendo dicho código con las instrucciones, en la colección de objetos "ThisWorkbook" (que también está considerado como un objeto) e indicando que ocurran cuando se abra (Open). Veámoslo paso a paso:

1) Creamos un nuevo libro de Excel e inmediatamente guardamos el archivo dándole el nombre que deseemos (en la imagen de abajo se llama "Libro1") y seleccionando el tipo de archivo: "Libro de Excel habilitado para macros". *Hay que recordar que si deseamos hacer este ejercicio en un archivo que ya teníamos, este debe estar guardado con el formato anteriormente mencionado (habilitado para macros).*

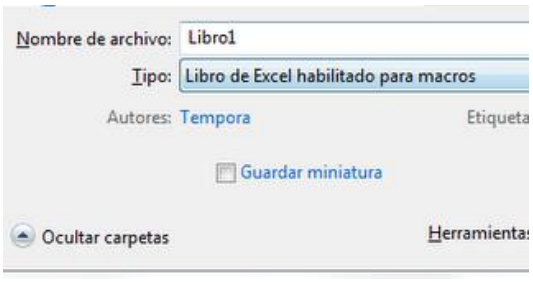

Ilustración 1

2) Hacemos click en la Ficha "Programador" y después en Visual Basic (a la izquierda). Con esto, se abre el editor de Visual Basic y deberíamos ver el panel "Explorador de proyectos" a la izquierda. (si no es así, podemos abrirlo con Ctrl+R o bien desde el comando "Explorador de proyectos" del menú "Ver").

3) Dependiendo de los complementos y archivos de Excel que tengamos habilitados y abiertos, veremos más o menos "Proyectos" en el panel abierto en el punto 1), pero entre ellos, debe estar el del archivo de Excel al que queremos asignar el código. Por ejemplo:

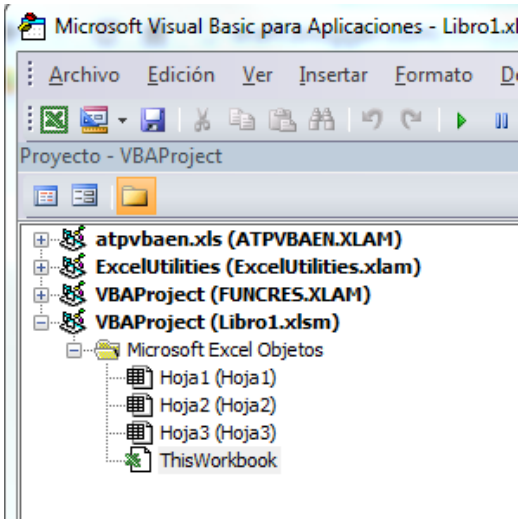

Ilustración 2

4) Como vemos en la imagen anterior, aparece la colección de objetos "**ThisWorkbook**", como un objeto más dentro de "Microsoft Excel Objetos". Pues debemos hacer doble click en "**ThisWorkbook**" para abrir a la derecha la **ventana de código** y seleccionar en el desplegable de la izquierda (Objetos) "**ThisWorkbook**" y en el desplegable de la derecha (Procedimientos) "Open", obteniendo lo siguiente:

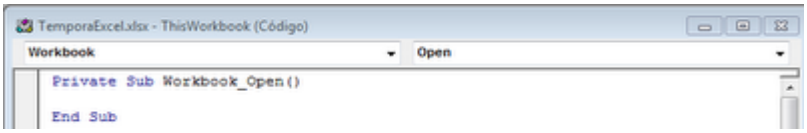

Ilustración 3

5) Vemos que **automáticamente se escriben las instrucciones Sub** y **End Sub** (además de otras palabras clave y parámetros) **como inicio y final del procedimiento**. Nuestro código debemos situarlo entre las dos líneas que se han generado automáticamente y por lo tanto, debe quedar todo así:

```
Private Sub Workbook_Open()
    Application.ScreenUpdating = False
    Sheets("Hoja3").Select
    Range("A1") = Now()
    Application.DisplayFullScreen = True
End Sub
```

6) **Cerramos el editor de Visual Basic, cerramos y guardamos el libro al salir y volvemos a abrir el libro**. Podemos comprobar que, automáticamente y sólo con la apertura del archivo, Excel selecciona la Hoja3 (Sheets("Hoja3").Select), escribe en la celda A1 la fecha y hora actual (Range("A1") = Now()) y además lo visualizamos a pantalla completa (Application.DisplayFullScreen = True). También hemos podido comprobar que, de todo el proceso, sólo vemos el resultado final gracias que Excel no va actualizando en pantalla lo que va procesando interiormente (Application.ScreenUpdating = False)

(Sólo tenemos que presionar ESC para salir del modo pantalla completa)

Esto es tan sólo un ejemplo sencillísimo, pero basta con escribir un código diferente diseñado con alguna finalidad y situarlo en el lugar explicado (entre las instrucciones Sub y End Sub), para obtener la automatización de una tarea.

A veces, es necesario saber el resultado que obtendríamos en un gráfico o en cualquier celda calculada, dependiendo de un valor que nosotros necesitamos variar de forma interactiva, dinámica e intuitiva. Esta necesidad suele surgir en planes de viabilidad, gestión de presupuestos y aplicaciones similares en las que la administración de varios escenarios es su razón de ser.

Ilustración 4

3 Automatizar una tarea cuando una celda cambia

3.1 Objetivo

En este artículo, **vamos a ver cómo podemos hacer que nuestra hoja de cálculo reaccione automáticamente** a una acción que nosotros deseamos, **cuando ocurre un EVENTO**. Concretamente, **veremos qué código debemos introducir y dónde, para conseguir que cuando una celda cambia de valor (EVENTO), se desencadene una acción elegida por nosotros... por ejemplo, que se muestre un mensaje al usuario**.

3.2 Solución propuesta

Empecemos por abrir nuestro explorador de proyectos. Empezamos por decir dónde hay que escribir el código. En el explorador de proyectos, seleccionamos la hoja donde está la celda que cambiará de valor y con el botón derecho, hacemos click en dicha hoja y seleccionamos "Ver código".

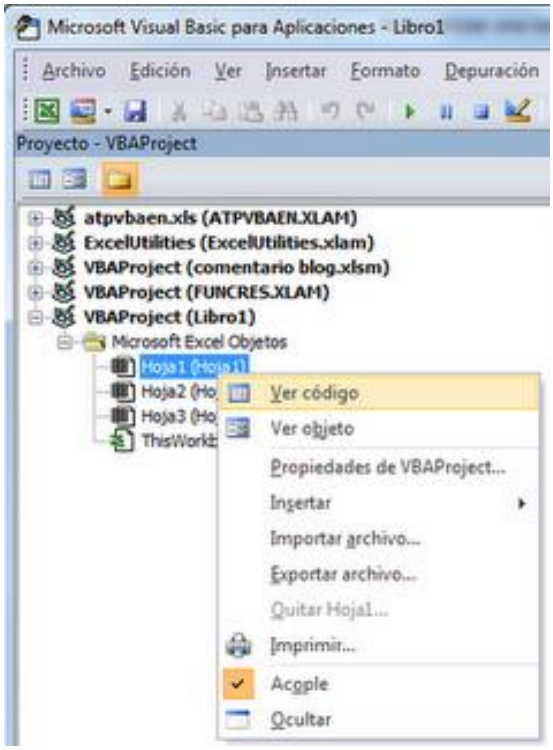

Ilustración 5

Se nos abre una ventana a la derecha y, en ella, debemos escribir el siguiente código:

```
Private Sub Worksheet_Change(ByVal Target As Range)
Celda = "B7"
If Not Application.Intersect(Target, Range(Celda)) Is Nothing Then
MsgBox ("Ha cambiado el valor de la celda")
End If
End Sub
```

Cerramos el editor de VBA y volvemos a la Hoja1 que es la que contiene la celda sobre la que hemos escrito el código (celda B7 .... como podemos leer dentro del código). Si escribimos algo para cambiar el valor de B7, obtendremos un mensaje:

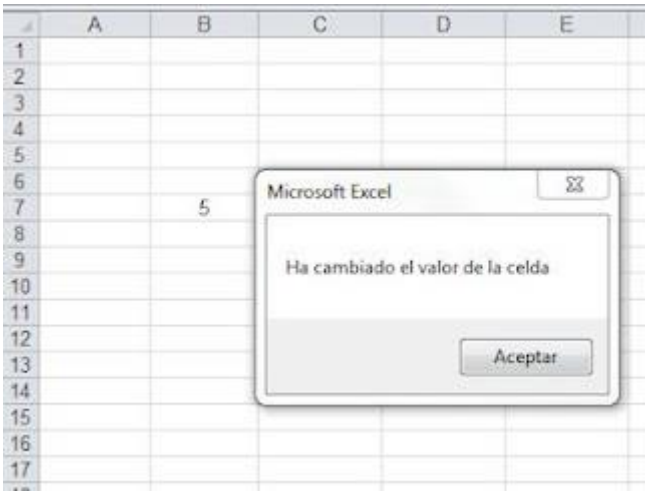

Ilustración 6

Basta con que sustituyamos la línea:

```
MsgBox ("Ha cambiado el valor de la celda")
```

por el código que queramos, y obtendremos una reacción al cambio sufrido por la celda B7.

También podemos hacer que la reacción se obtenga al cambiar cualquier cosa dentro de un rango de celdas (no sólo una celda). En ese caso, debemos sustituir en el código la referencia a la celda B7, por la referencia al rango de celdas (por ejemplo: B1:C4), quedando así:

```
Private Sub Worksheet_Change(ByVal Target As Range)
Rango = "B1:C4"
        If Not Application.Intersect(Target, Range(Rango)) Is Nothing Then
        MsgBox ("Ha cambiado el valor de la celda")
        End If
End Sub
```

4 Bibliografía

http://www.temporaexcel.blogspot.com.es/search/label/Nivel%203%3A%20Avanzado