Material Imprimible

Curso de Excel Macros

Módulo 2: Comenzando a programar en VBA

Programación de una macro

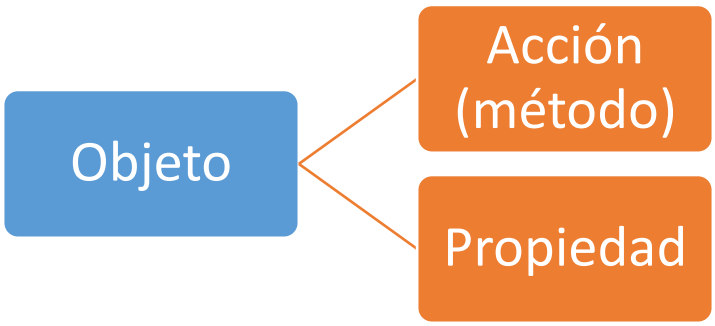
Sintaxis de un comando: Objetos. Propiedades y métodos

El lenguaje VBA es un lenguaje orientado a objetos, esto significa que cada elemento en Excel se representa en VBA como un objeto y que cada comando, instrucción u orden que escribamos en nuestra macro se va a escribir a partir de un objeto.

Cada objeto, además de tener una identidad (un nombre que lo hace único), tiene propiedades y métodos. Las propiedades representan las características del objeto y los métodos las acciones que se pueden llevar a cabo con él.

Existen sólo dos opciones para escribir un comando en Visual Basic:

- Mencionando una acción (o método) de un objeto.
- Estableciendo un valor a una propiedad de un objeto.



Sintaxis de un comando: Referencia a objetos

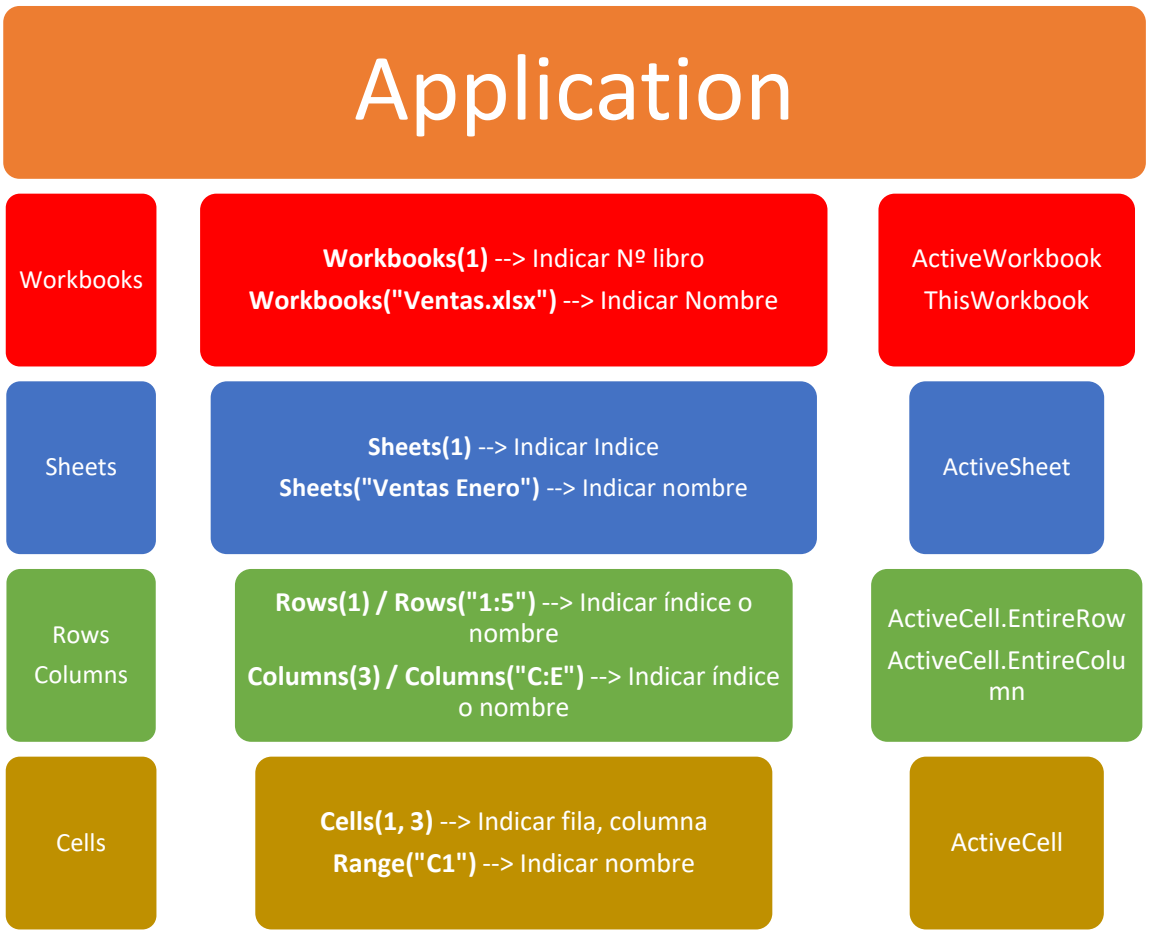
Existen varios objetos en Excel:

Application (aplicación de Excel), *Workbooks* (libros), *Sheets* (hojas), *Rows* (filas), *Columns* (Columnas), *Range* y *Cells* (celdas), entre otros.

Es necesario hacer referencia a ellos en cada comando de nuestra macro para luego indicar qué acción realizaremos o qué propiedad modificaremos.

Nos podemos referir a una colección de objetos o a un objeto en particular, dependiendo del comando que queramos indicar. Una colección es un conjunto de

todos los objetos del mismo tipo en Excel, por ejemplo, *Workbooks* es una colección de todos los libros que hay abiertos en Excel.



Application

El objeto *Application* representa a la aplicación misma de Excel. Sobre este objeto podemos definir una acción o una propiedad.

Ejemplo:

```
Application.Quit                    'Cierra la aplicación de Excel
```

Workbooks

Workbooks representa a la todos los libros de Excel.

Si queremos indicar una acción o una propiedad sobre la colección de libros, utilizaremos *Workbooks*. Pero si queremos indicar una acción o una propiedad sobre un libro en particular, tendremos que indicar a qué libro hacemos referencia (indicar

entre paréntesis el número o el nombre del libro – el número es según el orden en que fue abierto al ejecutarse la macro).

Ejemplos:

```
Workbooks.Open Filename:="C:LibroEjemplo.xlsx    'Abre un libro (de la colección)
Workbooks(1).Close                               'Cierra el libro 1
```

Por otra parte, el objeto *ActiveWorkbook* hace referencia al libro que actualmente se encuentra activo. *ThisWorkbook*, al que contiene el código de la macro que está en ejecución.

Sheets

Sheets representa a la todas las hojas de un libro de Excel.

Si queremos indicar una acción o una propiedad sobre la colección de hojas, utilizaremos SHEETS. Pero si queremos indicar una acción o una propiedad sobre una hoja en particular, tendremos que indicar a qué hoja hacemos referencia (indicar el índica o el nombre de la hoja – el índice es según la posición dentro del libro).

Ejemplos:

```
Sheets.Add                  'Agrega una hoja
Sheets(1).Activate          'Activa la hoja 1
```

Por otra parte, el objeto *ActiveSheet* hace referencia a la hoja que actualmente se encuentra activa.

Columns / Rows

Los objetos *Columns y Rows* representan una columna y una fila de una hoja de Excel respectivamente. A través de ellos podemos acceder a sus propiedades y acciones.

Para hacer referencia a columnas o filas específicas tendremos que escribir el rango correspondiente dentro del paréntesis o el número de fila o de columna.

Ejemplos:

```
Columns("E:E").Select       'Selecciona la columna E
Rows("1:10").Select         'Selecciona las filas 1 a 10
```

Columns(3).Select	'Selecciona la tercer columna (Columna C)
Rows(3).Select	'Selecciona la tercer fila (Fila 3)

Por otra parte, el objeto *ActiveCell.EntireRow* o *ActiveCell.EntireColumn* hace referencia respectivamente a la fila o columna de la celda activa. Se puede reemplazar el objeto *ActiveCell* por la celda o el rango que se desee.

Range / Cells

El objeto *Range* representa una celda, un rango o un conjunto de celdas en Excel. *Cells,* sólo a una celda individual. Es indistinto utilizar RANGE o CELLS para mencionar a una celda.

El objeto *Range* tiene como argumento, dentro de los paréntesis, el nombre de la celda o rango entre comillas dobles. Ahora bien, el objeto *Cells*, para las celdas, utiliza como argumentos el número de la fila y de la columna (en este orden) en que se halla la celda, separados por una coma.

Ejemplos:

Range("A2") .Select	'Selecciona la celda A2
Range("A1:J100") .Select	'Selecciona el rango A1:J100
Cells(2, 1) .Select	'Selecciona la celda A2 (fila 2, columna 1)

Por otra parte, el objeto *ActiveCell* hace referencia a la celda que actualmente se encuentra activa.

Sintaxis de un comando: Objetos y jerarquías

Los objetos en Excel están organizados jerárquicamente. El objeto con mayor jerarquía es *Application* (que representa a la aplicación de Excel), ubicándose los restantes por debajo de éste: *Workbooks* (libros), *Sheets* (hojas), *Rows* (filas), *Columns* (Columnas), *Range* y *Cells* (celda), entre otros.

Para acceder a los objetos que están por debajo de él utilizaremos el símbolo del PUNTO. De esa forma podremos navegar hacia jerarquías inferiores. También, a través

de este operador, podremos acceder a las distintas propiedades y métodos con las que cuentan estos objetos.

```
Application.ThisWorkbook.Worksheets(1).Range("A1").Font.Color = vbRed
```

En este ejemplo, hemos llegado al objeto *Range("A1")* (simbólico de la celda A1) a través del libro y de la hoja que la contienen (libro y hoja que no necesariamente son los activos). Posteriormente le definimos una propiedad: el color de la fuente.
Sin embargo, este comando que acabamos de escribir puede ser simplificado.
El objeto *Application* puede ser omitido puesto que Excel lo presumirá. Esta aplicación también asumirá que hacemos referencia al libro activo o a la hoja activa si omitimos referenciarlos en el código de VBA; pero si quisiéramos hacer referencia a otro libro u hoja estaremos obligados a escribirlos.

```
Range("A1").Font.Color = vbBlue
```

En este último ejemplo, se hace referencia al objeto celda A1 del libro y hoja activos (por estar omitidos los objetos).

Sintaxis de un comando: Acciones y propiedades

Todo comando en VBA se escribe a partir de un objeto, seguido de una acción o una propiedad a modificar. Tenemos que conocer cuáles son las acciones y las propiedades que se pueden indicar en cada uno de los objetos.
Para mencionar una acción o una propiedad sobre un objeto, la escribiremos a continuación de éste con un PUNTO separándolos.

Propiedades

Todos los objetos tienen distintas propiedades y cada una de estas propiedades un valor. Por ejemplo:

- la propiedad "color de relleno" de una celda tiene como valores posibles "azul", "blanco", etcétera.

- la propiedad “nombre” de una hoja tiene como valores posibles “enero”, “planilla” o el nombre que queramos indicarle.

A las propiedades, les asignaremos el valor que queremos indicarle con el símbolo de IGUAL.

Ejemplo:

A continuación, le asignamos el valor “Enero” al nombre – *name* – de la hoja 1 – *Sheets*(1) –.

```
Sheets(1).Name = “Enero”
```

Otros ejemplos:

Según la propiedad que estemos mencionando, los valores que se le pueden asignar pueden ser numéricos, de texto, lógicos o especiales. Solo si el valor es de texto se pondrá entre comillas.

En el caso de los valores lógicos, la opción True (verdadero) aplicará esa propiedad y False (falso) no la aplicará.

En el caso de los valores especiales, tendremos que conocer cuáles son las distintas posibilidades a aplicar.

```
Range("A1").Font.Size = 20
--> valor numérico (no va entre comillas)
Range("A1").Value = "Planilla de ventas"
--> valor texto (va entre comillas)
Range("A1").Font.Bold = True
--> valor lógico (True o False == aplico o no aplico esa propiedad)
Range("A1").Font.Color = vbBlue
--> los colores se escriben de FORMA ESPECIAL
Range("A1").Font.Underline = xlUnderlineStyleSingle
--> los tipos de subrayado se escriben de FORMA ESPECIAL
```

Acciones

Todos los objetos también tienen distintas acciones que pueden realizar.

Las acciones cuentan con parámetros para definir mejor qué es lo que harán y cada uno de estos parámetros puede adoptar un diferente valor. Los parámetros se escriben dejando un espacio a continuación de la acción. Para asignarles el valor que queremos a esos parámetros usaremos los símbolos DOS PUNTOS e IGUAL. Si se mencionan más de un parámetro, cada uno se separará con una COMA.

Ejemplo:

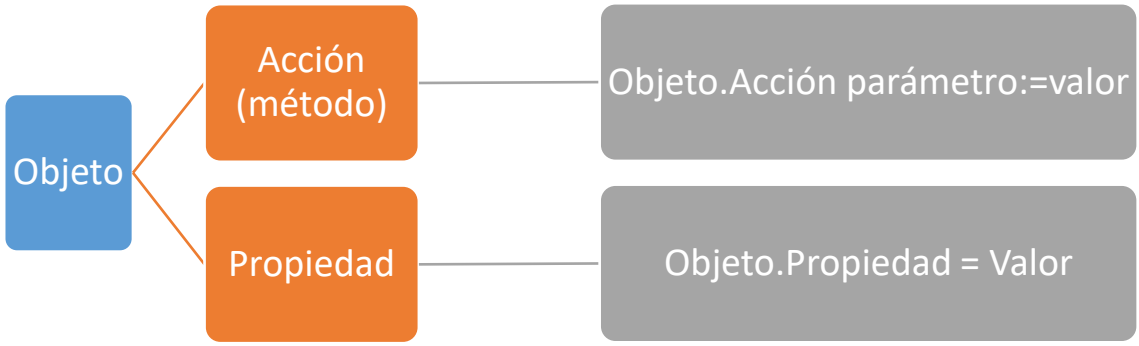
A continuación, con esta acción agregamos – *add* – una hoja – *Sheets* –.

Con el parámetro "after" le indicaremos en dónde se agregará esa nueva hoja.

Lo haremos después de la hoja activa – *ActiveSheet* –, según indica el valor del parámetro.

```
Sheets.Add After:=ActiveSheet
```

Resumen: sintaxis de un comando



Sintaxis de un comando: Ejemplos de acciones y propiedades sobre algunos objetos

Ejemplo con APPLICATION:

```
Application.Quit                    'Cierra la aplicación de Excel
```

Ejemplos con WORKBOOKS:

```
Workbooks.Open Filename:="C:LibroEjemplo.xlsx"   'Abre un libro (de la colección)
Workbooks(1).Save                   'Guarda el libro 1
Workbooks(1).Close                  'Cierra el libro 1
```

Ejemplos con SHEETS:

```
Sheets.Add                          'Agrega una hoja
Sheets(1).Activate                  'Activa la hoja 1
Sheets(1).Copy After:=Sheets(3)     'Copia la hoja 1 luego de la hoja 3
Sheets(1).Move After:=Sheets(3)     'Mueve la hoja 1 luego de la hoja 3
Sheets(1).Delete                    'Elimina la hoja 1
Sheets(1).Tab.Color = vbRed         'Cambia el color de la etiqueta de la hoja 1 a rojo
Sheets(1).Visible = False           'Cambia el estado de la hoja a oculta
```

Ejemplos con ROWS/COLUMNS:

```
Columns("E:E").Select               'Selecciona la columna E
Rows("1:10").Select                 'Selecciona las filas 1 a 10
Cells.EntireColumn.ColumnWidth = 20 'Ancho de columna de todas las celdas: 20
Cells.EntireRow.RowHeight = 20      'Alto de fila de todas las celdas: 20
Cells.EntireColumn.Autofit          'Autoajusta las columnas de todas las celdas
```

Ejemplos con RANGE (también aplica para CELLS):

```
Range("A1").Activate                'Activa la celda A1
Range("A1").Select                  'Selecciona la celda A1
Range("A1").Font.Color = vbRed      'Cambia el color de fuente de la celda A1 a rojo
```

```
Range(“A1”).Font.Size = 20              ‘Cambia el tamaño de fuente de la celda A1 a 20
Range(“A1”).Font.Name = Arial           ‘Cambia el tipo de fuente de la celda A1 a Arial
Range(“A1”).Font.Bold = True                ‘Cambia la fuente de la celda A1 a Negrita
Range(“A1”).Font.Italic = True          ‘Cambia la fuente de la celda A1 a Cursiva
Range(“A1”).Interior.Color = vbRed ‘Cambia el color de relleno de la celda A1 a rojo
Range(“A1”).Borders.Color = vbRed ‘Cambia el color de bordes de la celda A1 a rojo
Range("A1").Font.Underline = xlUnderlineStyleSingle      ‘Aplica subrayado simple
```

Sintaxis de un comando: Uso de la estructura ‘WITH’.

El bloque *With* nos sirve para evitar hacer referencia en forma reiterada a un mismo objeto en varias líneas cuando definimos varias propiedades o métodos. Este bloque inicia con el comando *With* y finaliza con *End With*. Todo lo que se encuentre entre esas dos líneas compartirá el objeto inicial, aunque no se lo vuelva a escribir.

En vez de escribir…

```
Application.ThisWorkbook.Worksheets(1).Range(“A1”).Value = “Planilla de Enero”
Application.ThisWorkbook.Worksheets(1).Range(“A1”).Font.ColorIndex = 1
Application.ThisWorkbook.Worksheets(1).Range(“A1”).Font.Bold = True
```

…podremos referenciarnos al objeto una única vez y establecer sus propiedades dentro del bloque:

```
With Application.ThisWorkbook.Worksheets(1) .Range(“A1”)
   .Value = “Planilla de Enero”
   .Font.ColorIndex = 1
   .Font.Bold = True
End With
```

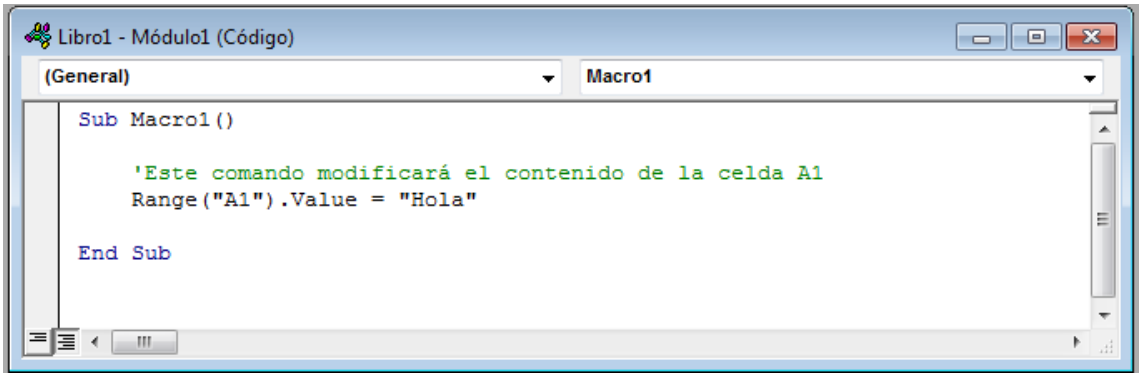
Sintaxis de un comando: Los comentarios en VBA

Un comentario en VBA es una línea de código que no será ejecutada al ejecutar la macro. Serán sólo visibles en el editor de Visual Basic. Normalmente se los suele utilizar para entender qué se pretende realizar con una determinada cadena de comandos o

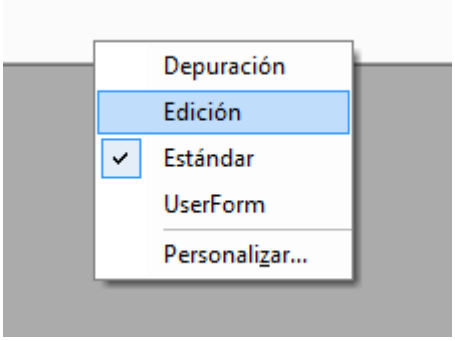
su lógica o para agregar información que será utilizada más adelante. También es útil cuando deseamos ejecutar la macro parcialmente para probarla y aún no hemos terminado de escribir ciertos comandos (tendremos que poner como comentarios a estos últimos para que no se ejecuten y nos aparezca un error).

La línea de comentario en el editor se visualiza en color verde. Ello implica que, aunque allí haya una instrucción, la misma no será puesta en ejecución.

Para agregar un comentario sólo es necesario agregar un apóstrofe o comilla simple (') al iniciar la línea.



Si quisiéramos convertir en comentario a varias líneas de una vez, podremos seleccionar el bloque de líneas y presionar el botón . Para convertir a ese bloque nuevamente en código ejecutable tendremos que seleccionar el botón . Ambos botones agregan o quitan un apóstrofe al principio de la línea cada vez que son presionados.



Los botones mencionados se encuentran en la barra de edición del editor de VBA. De no estar activa esta barra tendremos que hacer click derecho con el mouse en cualquiera de las barras ubicadas en la parte superior de la ventana y activarla seleccionándola con un click izquierdo.

Tipos de errores que se presentan

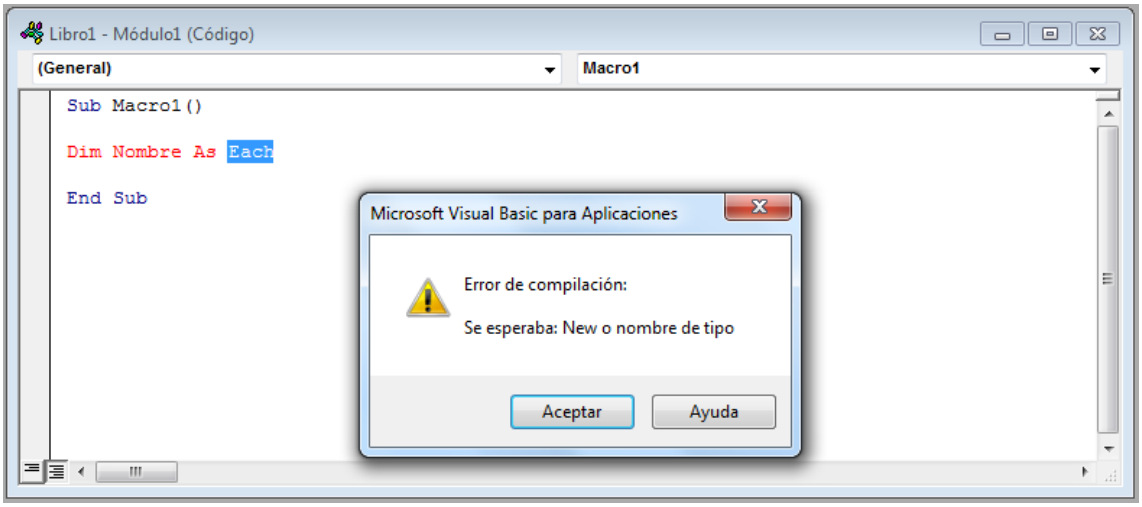
Los errores en VBA impiden que se ejecuten las tareas programadas para una macro. Cuando se detecta un error la macro automáticamente detiene su ejecución puesto

que no puede continuar. Para evitar que ello suceda se debería utilizar el depurador previamente.

Existen dos tipos de errores: los errores de sintaxis y los errores en tiempo de ejecución.

- Los errores de sintaxis: Son aquellos que se producen en la escritura misma de la instrucción en el lenguaje VBA. Ocurren cuando intentamos insertar algún operador o alguna instrucción en un lugar que no corresponden o que no que no son válidos.

  Este tipo de errores es fácilmente detectable puesto que el editor de Visual Basic nos alertará con un mensaje de error. Asimismo, el color de la fuente del código se tornará rojo.

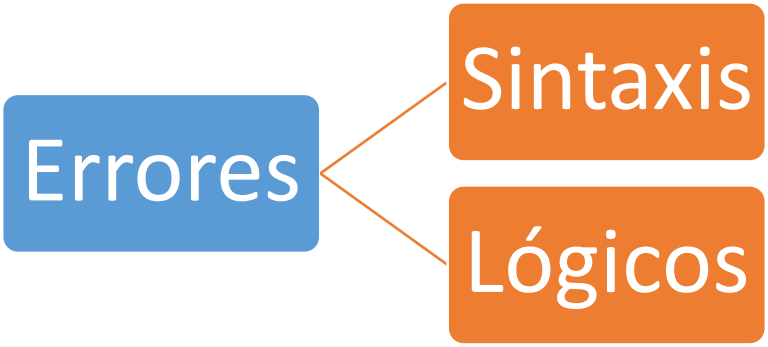


En esta imagen se intentó declarar la variable "Nombre", pero cuando se esperaba un tipo de variable válida, como ser "*string*" o "*Integer*", se insertó un tipo de variable incorrecto (pese a ser un comando válido en otro contexto).

- Los errores en tiempo de ejecución: Son aquellos que se producen cuando la aplicación ya está siendo ejecutada y se intenta realizar una acción no permitida por Excel o por el sistema operativo, lo cual provocará que Excel deje de responder.

Este tipo de error es más difícil de detectar aun utilizando la opción de depuración.

Un ejemplo de este tipo de error sería utilizar un bucle con una condición que nunca se cumple, puesto que las acciones contenidas en él se ejecutarán indefinidamente.



Depurar una macro

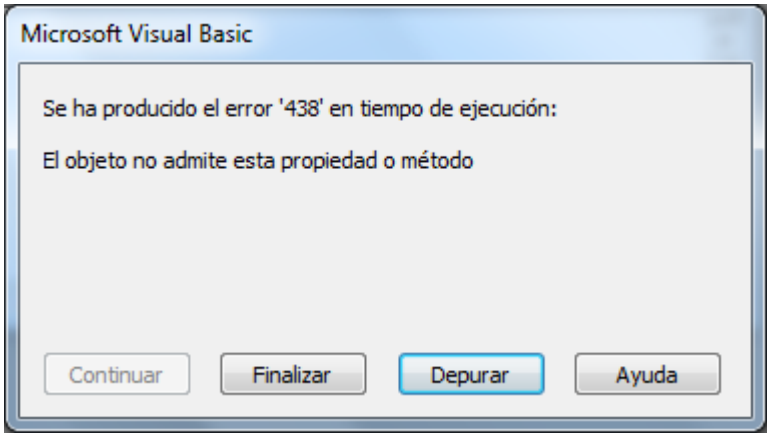
Depurar una macro implica probarla ejecutándola con el objetivo de encontrar errores y eliminarlos, si existieren.

Nota: Cuando ejecute el depurador las instrucciones se ejecutarán realmente, por tanto, su planilla de Excel sufrirá las acciones que se le indiquen.

Para iniciar la depuración de una macro tendremos que ir al menú Depuración de la Barra de menús y seleccionar la opción Paso a paso por instrucciones o simplemente apretar la tecla F8, siempre teniendo en cuenta que debemos posicionarnos con el cursor del mouse en cualquier parte del código de la misma.

Si no estuviéramos en el editor de Visual Basic podemos abrir el listado de Macros disponibles en el libro (mediante la opción Macros de la pestaña Programador o Desarrollador o Vista de la cinta de opciones), seleccionar la macro a depurar y presionar el botón Paso a Paso.

Iniciando la depuración, se resaltará en amarillo la primera línea de código de nuestra macro a testear. Esto indica que la macro será ejecutada línea a línea (paso a paso). Por tanto, para ir probando la ejecución de la macro habrá que volver a seleccionar la opción "Paso a paso por instrucciones" o la tecla F8 hasta llegar al final del código. Al hacer esto, Excel ejecutará la instrucción seleccionada en amarillo. Si hubiera un error lo notificará; caso



contrario, resaltará en amarillo la siguiente línea de código. Cuando se detecte un error (independientemente de cuál) aparecerá un cuadro de diálogo indicando que se ha producido un error en tiempo de ejecución e indicará de qué tipo de error se trata para poder corregirlo y eliminarlo. Al presionar el botón "Finalizar", se cerrará el cuadro de diálogo finalizando de esa forma la depuración. Si presionáramos el botón "Depurar", al cerrarse el cuadro de diálogo seguirá la depuración, aunque podremos ya corregir la instrucción y volver a probarla. La opción de "Ayuda" nos brindará la posibilidad de obtener asistencia en línea acerca del error ocurrido.

Podremos finalizar la depuración de nuestra macro en cualquier momento presionando el botón restablecer  de la barra estándar del Editor.

Manejo de errores

Con el comando "ON ERROR" podremos manejar qué hacer en caso de error en una macro, evitando que aparezca el error de VBA.

Al indicar ON ERROR RESUME NEXT, le indicamos a la macro que, en caso de error lo omita y siga ejecutando el resto de la macro, si es posible.

Al indicar ON ERROR GOTO ETIQUETA, le indicaremos a la macro que, en caso de error realice un comando o conjunto de comandos especificados en un bloque llamado ETIQUETA. Podremos ponerle a ETIQUETA cualquier nombre (sin espacios intermedios).

En este último caso, tendremos que definir qué hará esa ETIQUETA al final de nuestra macro. Escribiremos nuevamente el nombre de la ETIQUETA seguida de dos puntos. Debajo escribiremos el o los comandos que forman parte de la misma.

Nota: Es importante que antes de definir la etiqueta escribamos EXIT SUB (para finalizar la macro) ya que los comandos que se van a mencionar a continuación sólo deberán ocurrir si se produce el error.

Ejemplo:

```
Sub AplicarFiltro()

'Verifica si la celda seleccionada está vacía. En ese caso, devuelve un mensaje de error
On Error GoTo ErrorFiltro

    'Activa o desactiva el filtro de la selección
    ActiveCell.AutoFilter

    'Finaliza la macro
    Exit Sub

'Etiqueta creada para indicar un cuadro de mensaje si ocurre un error.
ErrorFiltro:
MsgBox "Debe seleccionar una celda de una tabla".

End Sub
```