Apuntes Programación Excel VBA II

Introducción a la programación VBA con Excel

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

INDICE

1 Fundamentos VBA 1
1.1 Introducción 1
1.2 Teclas de Acceso Rápido del Editor VBA 1
1.3 Variables VBA. 1
1.3.1 Introducción 1
1.3.2 ¿Es conveniente usar la instrucción Option Explicit al principio de los módulos? 1
1.3.3 ¿Qué nivel de seguridad es el más aconsejable en lo relativo a la ejecución de código? 2
2 Introducción a las macros y eventos 3
2.1 Macros 3
2.1.1 Objetivos de las Macros 3
2.1.2 El menú general de macros, primera aproximación 4
2.1.3 Contenedores de Macros 5
2.1.4 Creación y grabación de una macro. Definición de las acciones a desarrollar por la macro. 5
2.1.5 Ejecución de una macro 6
2.1.6 Análisis del código generado por la macro. Editor de Visual Basic. 7
2.1.7 En la ilustración anterior disponemos del Editor de Visual Basic que nos permitirá entre otras cosas analizar las macros. 8
2.1.8 Las limitaciones del grabador de macros 9
2.1.9 Donde guardar las macros. Este libro, libro nuevo o libro de macros personales. 9
2.2 Eventos 10
2.2.1 Introducción 10
2.2.2 Nombre de los Eventos en Excel. 10
2.2.3 Eventos de una Hoja Específica de un Libro Excel. 11
2.2.4 Eventos de un Libro Excel 11

2.2.5 Resumen: Objetos, Eventos de libro y eventos de hoja 12
2.3 Ejecutar una macro a través de un botón o vinculada a una celda 12
2.3.1 Asignar una macro a un botón 12
2.3.2 Asignar diferentes macros a diferentes celdas 12
2.3.3 Caso especial asignar varias macros a varias celdas al activar o hacer doble click 14
3 Los módulos 15
3.1 Introducción, ¿qué es un módulo? 15
3.2 Cómo acceder a un módulo de VBA y como crearlo 15
3.3 Ejemplo de módulo de VBA 16
3.4 Como llamar a un procedimiento Sub dentro de un módulo. Instrucción Call y Método Run 16
3.4.1 Ejecutar macros de otro libro de Excel con el método Run 16
3.4.2 Ejecutar macros de otro módulo del mismo archivo con la instrucción Call 17
3.4.3 Ejemplo 17
4 Bibliografía 18

1 Fundamentos VBA

1.1 *Introducción*

Cada producto Excel, Word, etc tiene su propio modelos de objetos únicos. Mientras trabajamos con excel iremos comprendiendo gradualmente el modelo de objetos. Puede ser muy complicado de entender al principio, pero al final se comprendera perfectamente.

Código:
Se escribe o se graba código VBA que se almacena en un módulo VBA.

Módulos:
Consiste en todo un conjunto de procedimientos. Los modulos VBA estan almacenados en un libro de Excel y se editan con el editor VBE

Procedimientos:
Es una unidad de código que realiza akguna acción

Ilustración 1

1.2 *Teclas de Acceso Rápido del Editor VBA.*

Teclas Rápidas del Editor de Visual Basic

Alt + F11	Editor Visual Basic (y volver a Excel)	F5	Ejecutar Macro
Ctrl + R	Ver Explorador de Proyectos		
F4	Ver Ventana de Propiedades		**Depuración**
F7	Ver Código de un Objeto (Formulario)	F8	Ejecuta Paso a Paso
May + F7	Ver Objeto (Formulario)	May + F8	Paso a paso por Procedim.
Ctrl + J	Lista de Propiedades y Métodos	Ctrl + F8	Ejecuta hasta el Cursor
Ctrl + Esp	Completa la palabra	F9	Punto de Interrupción
F2	Examinador de Objetos		
May + F2	Ver Definición		
F1	Ayuda sobre la palabra seleccionada	Ctrl + S	Guardar Proyecto

1.3 *Variables VBA.*

1.3.1 Introducción.

Variables VBA:
Podemos asignar valores a variables VBA. Piensa en una variable como en un nombre que puedes utilizar para almacenar un determinado valor. Para asignar el valor de la celda A1 de la Hoja1 a una variable llamada Precio, utiliza la siguiente instrucción VBA: Precio = Worksheets ("Hoja1").Range("A1").Value

1.3.2 ¿Es conveniente usar la instrucción Option Explicit al principio de los módulos?

Sí. Y además es aconsejable configurar el editor de VBA para que siempre ponga automáticamente dicha instrucción en todos los módulos nuevos. Esto se hace desde Herramientas->Opciones->Solapa "Editor", marcando la casilla "Requerir declaración de variables".

Esta instrucción obliga al programador a declarar las variables de forma explícita, lo que en principio puede resultar molesto pero a la larga facilita la depuración del código y evita errores muy difíciles de localizar, especialmente en proyectos grandes.

1.3.3 ¿Qué nivel de seguridad es el más aconsejable en lo relativo a la ejecución de código?

El nivel de seguridad para la ejecución de código por defecto (Herramientas->Macro->Seguridad) está en "Medio", lo que significa que antes de abrir un libro cuyo código no está firmado por una fuente de confianza Excel pregunta si se autoriza o no su ejecución; y lo mejor es dejarlo así.

Rebajarlo a la seguridad "Mínima" no acarrearía ningún riesgo si jamás se fuera a abrir un libro creado por un tercero, lo que hoy en día, con el correo electrónico y el trabajo en grupo, resulta muy difícil garantizar. Y aumentar la seguridad a su nivel máximo impediría la ejecución de cualquier código, a no ser que el proyecto VBA estuviera firmado por una fuente de confianza.

2 Introducción a las macros y eventos

2.1 *Macros*

2.1.1 Objetivos de las Macros

Una macro es una combinación de instrucciones que pueden ser ejecutadas automáticamente con una simple pulsación de teclas. La palabra macro es una abreviatura de la palabra macroinstrucción que viene a ser lo mismo que hemos definido. En ocasiones, nos vemos en la necesidad de realizar una serie de tareas repetitivas de forma rutinaria. Podemos crear una macro que nos evite ese trabajo. Una macro en sí es un pequeño programa en código visual basic que se graba con un nombre y que podemos invocar en cualquier momento. También podemos asignar una combinación de teclas como Control+V para invocarla. La ejecución de una macro es muy rápida, aunque depende de la cantidad de instrucciones que deba realizar.

Imaginemos que diariamente tenemos que arreglar una tabla de datos poniéndole colorines, formato, bordes, etc para posteriormente imprimirla. La ilustración de la izquierda muestra la tabla normal, y la de la derecha arreglada a mano:

	A	B	C
1	3454	1234	12341
2	4353	2342	21341
3	5433	2331	123412
4	4634	12341	23423
5	4333	23432	12342
6	4533	213341	213412

	A	B	C
1	3.454 Pts	1.234 Pts	12.341 Pts
2	4.353 Pts	2.342 Pts	21.341 Pts
3	5.433 Pts	2.331 Pts	123.412 Pts
4	4.634 Pts	12.341 Pts	23.423 Pts
5	4.333 Pts	23.432 Pts	12.342 Pts
6	4.533 Pts	213.341 Pts	213.412 Pts

Ilustración 2

Para arreglar la tabla tendríamos que:

1. Seleccionarla
2. Abrir la paleta de bordes y colocar un borde exterior
3. Abrir la paleta de color de fondo y escoger un color
4. Abrir la paleta de color de texto y escoger un color
5. Pulsar un click en el botón del formato monetario
6. Pulsar un click en el botón del forlmato cursiva
7. Pulsar un click fuera de la tabla y extasiarnos con su belleza

Estos han sido sólo 7 pasos. Imagina una tarea rutinaria de 200 pasos. Para ello, podemos crear una macro que nos realice el trabajo automáticamente. Evidentemente, los pasos de la macro han de ser los correctos, evitando abrir y cerrar menús y opciones innecesariamente, por lo que la macro tardaría más en ejecutarse.

2.1.2 El menú general de macros, primera aproximación

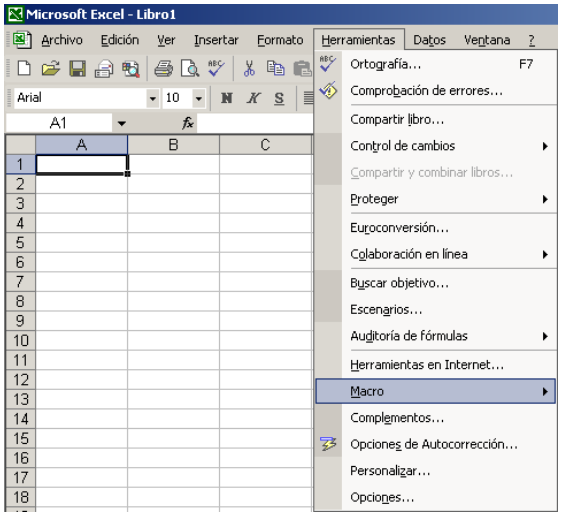

Ilustración 3

Para aproximarnos por primera vez al estudio de VBA para Excel una buena opción será el estudio y comprensión de las Macros de Excel.

Una macro no es más que una serie de comandos guardados con un nombre que Excel puede ejecutar.

Para acceder a la opción de macros tendremos que ir al menú de Herramientas de Excel y pulsar sobre la opción Macros (tal y como se refleja en la Ilustración 3). De esta forma podremos acceder al menú principal de todas las opciones relacionadas con las macros de excel.

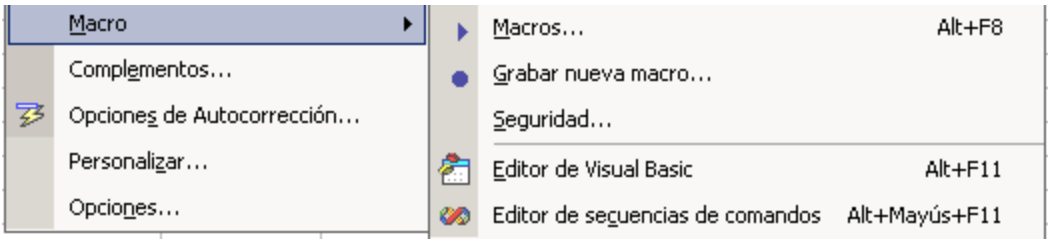

Ilustración 4

En este Menú General de Macros las opciones que se nos presenta y su significado es el siguiente:

- Macros. Nos permite acceder a todas las macros disponibles y ejecutarlas.
- Grabar nueva macro. Como su propio nombre indica permite acceder a la grabación de macros.
- Seguridad. Excel incluye protecciones contra virus susceptibles de ser transmitidas por macros. Si se comparten macros con otros usuarios, se puede certificar esas macros con una firma digital de modo que los demas usuarios pueden comprobar que proceden de una fuente fidedigna. Al abrir un libro que contiene macros, se puede comprobar su origen antes de habilitarlas.
- Editor de Visual Basic. Tras grabar una macro, se puede ver el código de macro con el Editor de Visual Basic para corregir errores o modificar lo que hace la macro. El Editor de Visual Basic es un programa diseñado para que los usuarios principiantes puedan escribir y editar fácilmente código de macro, y proporciona mucha Ayuda en pantalla. No es preciso saber cómo se programa o se utiliza el lenguaje de Visual Basic para realizar cambios sencillos en las macros. El Editor de Visual Basic permite modificar macros, copiarlas de un módulo a otro, copiarlas entre diferentes libros, cambiar el nombre de los módulos que almacenan las macros o cambiar el nombre de las macros.
- Editor de Secuencias de Comandos.

En esta primera toma de contacto lo que nos interesa básicamente es conceptualizar los aspectos fundamentales relacionados con las Macros y en esta linea debemos tener en cuenta que el grabador de macros permite grabar una serie de acciones a las que convierte en código VBA. Normalmente los grandes expertos en VBA siguen usando el grabador de macros para el desarrollo de sus aplicaciones ya que el grabador de macros ahorra mucho tiempo en términos de programación.

2.1.3 Contenedores de Macros

Un libro de Excel tiene por lo menos 3 hojas, entonces, existen mínimo 4 POSIBLES contenedores para las Macros, una por cada hoja y un contenedor por el libro en sí.

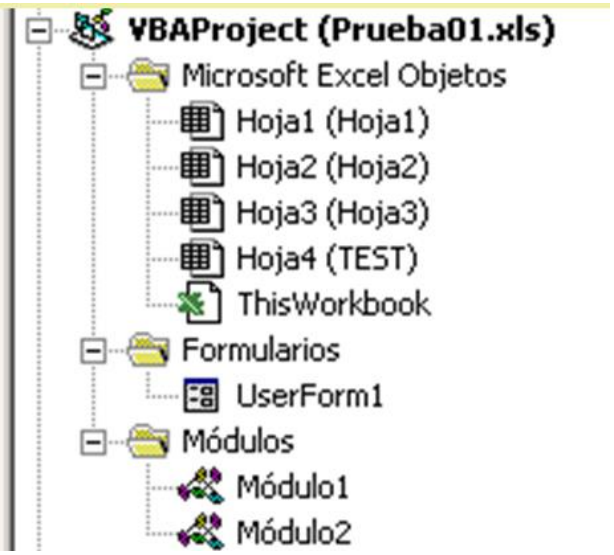

Ilustración 5

2.1.4 Creación y grabación de una macro. Definición de las acciones a desarrollar por la macro.

Una buena forma de tomar contacto con las macros es desarrollar nuestra primera macro.

Ejemplo 1.- Nuestra primera Macro. Fuente_grande_rojo.

Objetivo.

Deseamos crear una macro que cambie la fuente y el color de la celda seleccionada. En un libro nuevo de Excel realizaremos las siguientes operaciones

1. En la celda A1 Introducimos nuestro nombre, en la celda B1 Apellido, en la celda C1 nuestra ciudad y en la D1 el país. Tal y como aparece en la siguiente ilustración.

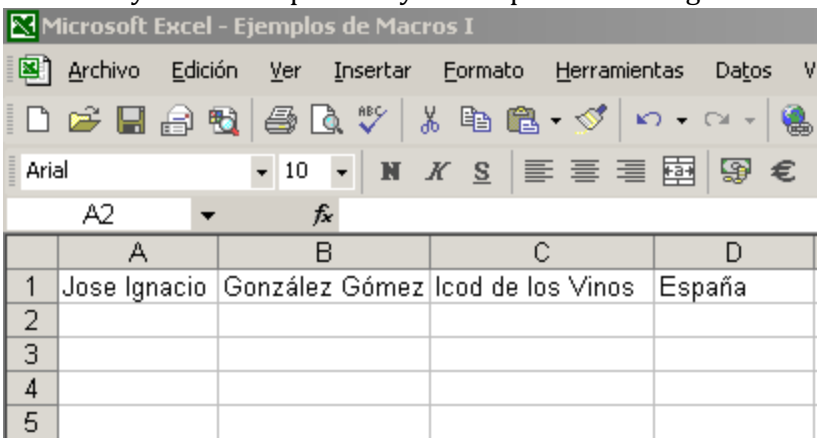

Ilustración 6

2. Nos situamos sobre la celda A1 y accedemos al menú grabar (Herramientas-Macros) y seleccionamos la opción Grabar Macro con lo que aparece el cuadro de dialogo correspondiente a la citada opción.

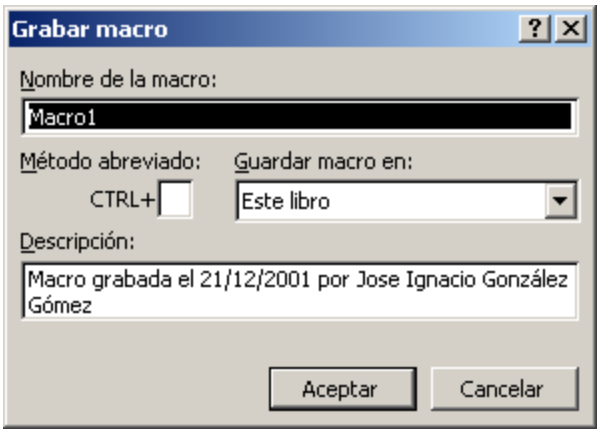

Ilustración 7

3. Las opciones del cuadro de dialogo de Grabar macro son las siguientes:
    a. Nombre de la macro. Seleccionaremos el nombre que deseamos asignar a la macro por ejemplo para nuestro caso Fuente_grande_rojo. El nombre de una macro puede ser de hasta 255 caracteres de longitud y debe empezar con una letra. Se permiten números, guiones o caracteres de guion bajo pero no espacios. Normalmente se usa un carácter de guion bajo para representar los espacios.
    b. Ctr + . Nos permite definir el metódo abreviado para acceder a la macro.
    c. Gaurdar macro en:. Nos permite establecer si la macro se guardara en este libro o estará disponible cada vez que abramos el excel es decir para todos los libros. En este caso solo estara disponible en el libro que hemos abierto.
    d. Descripción. Permite establecer una descripción mas amplia de la macro.
4. Definición de las acciones que desarrollará la macro. Al aceptar la cración de macro en la Ilustración 7 comenzaremos a definir las acciones correspondientes que hará la macro. En concreto serán dos:
    a. Estableceremos como tamaño de la fuente Ariel16
    b. Color de la fuente rojo.

    Para ello vamos al menu de Excel en la opción formato de celda y seleccionamos los citados valores.

5. Finalización de la grabación de la macro. Para terminar de grabar la macro debemos de pulsar el botón detener que se encuentra activado en la opción Herramientas-Macros.

2.1.5 Ejecución de una macro

Cuando ejecutamos una macro, esta lleva a cabo los mismos pasos que realizamos durante su grabación. En este caso para ejecutar la macro Fuente_grande_roja creada anteriormente y aplicarla sobre la celda B1 tendremos que posicionarnos sobre la citada celda y seleccionando la macro prevista (Herramientas – Macros) pulsamos sobre el botón ejecutar que aparece en la pantalla de dialogo de macros tal y como podemos observar en la Ilustración 8.

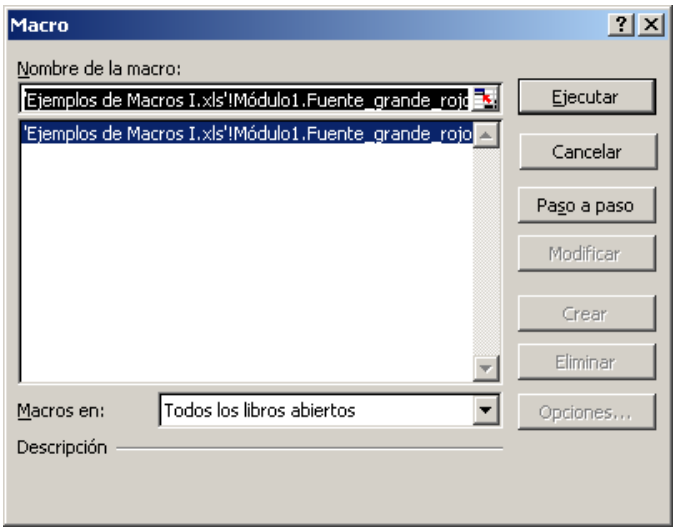

Ilustración 8

De igual forma procederemos con el resto de celdas para aplicar el formato seleccionado.

2.1.6 Análisis del código generado por la macro. Editor de Visual Basic.

Acceso al código de la macro. El editor de visual basic.

Mientras hemos creado la macro anterior y en concreto definimos las distintas acciones a desarrollar por la misma (cambiar tamaño de la fuente a 16 y cambiar el color de la fuente a rojo) Excel convirtió esas acciones en código VBA.

Si quisiéramos analizar el mismo solo tendriamos que ir al menú Herramientas – Macros seleccionar la deseada (Fuente_grande_roja) y pulsar sobre el botón Modificar con lo que se abrirá el editor Visual Basic que nos permitirá analizar cómo se ha confeccionado la citada macro.

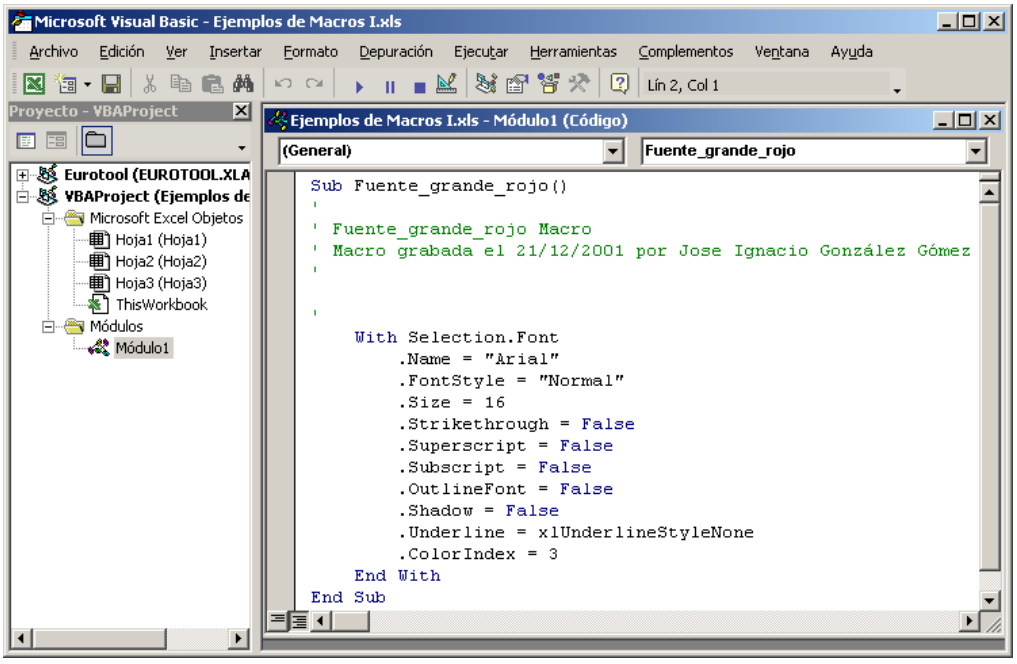

Ilustración 9

2.1.7 En la ilustración anterior disponemos del Editor de Visual Basic que nos permitirá entre otras cosas analizar las macros.

Análisis del procedimiento Fuente grande rojo.

En concreto en este caso los que nos interesa analizar es la parte derecha de la Ilustración 9 que denominaremos procedimiento Fuente_grande_rojo. La primera línea de código Sub Fuente_grande_rojo ( ), representa el punto de inicio y el nombre de la macro.
Las líneas que siguen y comienzan con un apostrofe ´ son comentarios que representan documentación acerca de la macro; en este caso el nombre de la macro y cuándo y por quien fue grabada.

La parte que realmente trabaja comienza con la palabra With y la palabra Selection, es una palabra que utiliza VBA para referirse a las celdas que estén seleccionadas en la hoja de cálculo. Esta es la razón por la cual la macro trabaja este una celda o varias seleccionadas. También podemos observar que debajo de la línea anterior aparecen las propiedades que hemos seleccionado para la macro en concreto el tamaño y el color conjuntamente con otra serie de propiedades que no hemos modificado pero que están asociadas a la ficha fuente de formato de celda y por tanto aparecen aunque la misma no se ha modificado.

Modificación de una macro. Editor de Visual Basic.

Existen muchas razones por las cuales podríamos necesitar editar una macro, por ejemplo corregir algún error que hubiéramos cometido durante la grabación de la macro o bien modificar algunos de sus parámetros.
La modificación del código de una macro se hace directamente en el Editor de Visual Basic, pudiendo agregar, suprimir y modificar líneas. Por ejemplo y para el caso anterior (ver Ilustración 9) podríamos suprimir aquellas líneas que no actúan sobre la macro en cuestión es decir aquellos campos de la ficha fuente formato de celda que se han grabado por defecto pero que no afectan a las actividades o acciones a realizar por nuestra macro, solo vamos aumentar el tamaño de la fuente a 18. Esta acción no modificará en absoluto el comportamiento de la macro y el resultado de tal acción aparecerá reflejado a continuación.

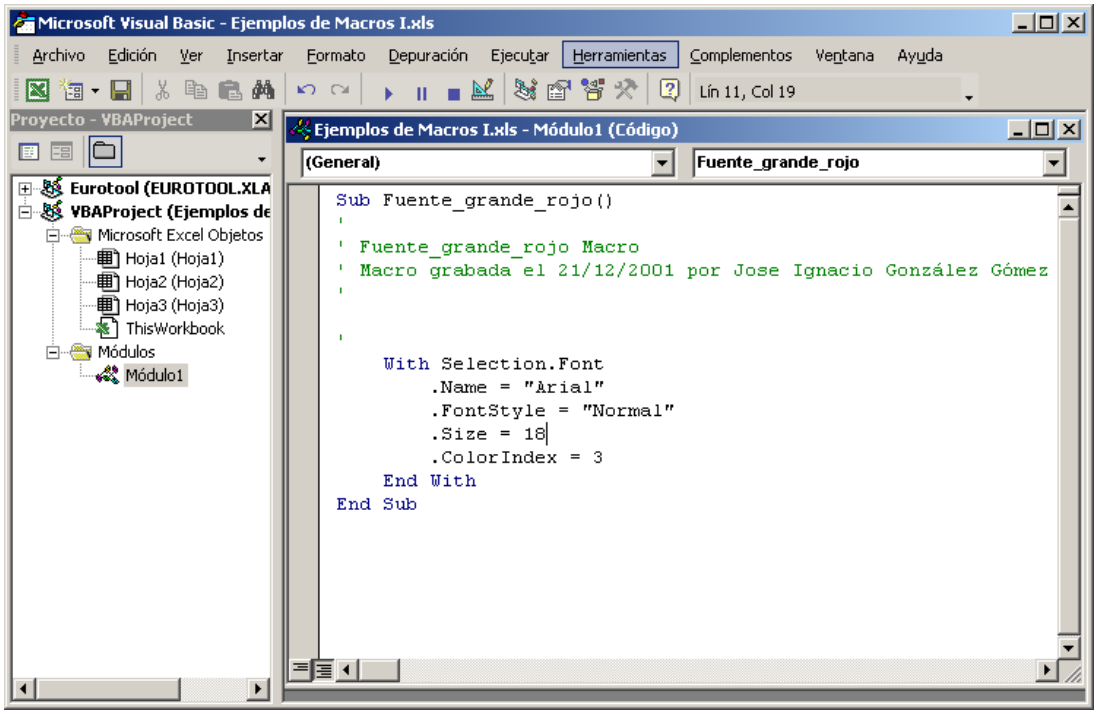

Ilustración 10

Pulsando sobre el icono grabar de la ilustración anterior la macro quedará grabada.

2.1.8 Las limitaciones del grabador de macros.

Muchos de los procesos de Excel que necesitamos automatizar pueden lograrse mediante la grabación de sus acciones, a través de las macros. Pero el grabador de macros tiene limitaciones.
Entre las limitaciones que el grabador de macros tiene podemos señalar:

- Pedir información al usuario mientras se está ejecutando la macro.
- Ejecutar diferentes acciones basadas en los datos proporcionados por el usuario o en el valor de las celdas.
- Desplegar cuadros de dialogo de Excel, por ejemplo el cuadro de dialogo Guardar como.
- Desplegar y emplear formularios personalizados.

Estas limitaciones son solo algunas de las razones por las que se necesitará crear nuestro propio código VBA.

2.1.9 Donde guardar las macros. Este libro, libro nuevo o libro de macros personales.

Cuando creamos nuestra primera macro, normalmente aceptamos la ubicación predeterminada para que la misma sea guardada. En realidad existen tres posibilidades donde guardar nuestras macros:

1. Este libro
2. Libro nuevo
3. Libro de macros personales

Cuando elegimos la primera opción para guardar la macro esta reside exclusivamente en el libro actual y solo estará disponible para el citado libro Excel.

También es posible guardar nuestra macro en un libro nuevo con lo que el mismo se creará automáticamente. La opción final es guardarlo como libro de macros personales.

El libro de macros personales es un libro especial oculto que se reserva solo para almacenar las macros. La primera vez que elegimos guardar una macro en este libro, se crea un archivo nuevo llamado PERSONAL.XLS y que está guardado en la carpeta XLSTART.

Después de que el archivo este creado cada vez que abrimos un nuevo libro de excel automáticamente se abre el mismo para tener a disposición todas las macros que hemos creado y guardado en el citado documento. Esto significa que si creamos una macro que sea lo bastante genérica para poder ser usada por múltiples libros, necesitaremos que sea guardada en este libro de macros personales.

Para guardar una macro en el libro de macros personales procederemos de forma habitual a crear nuestra macro solo que a la hora de seleccionar Guardar como elegiremos el destino deseado tal y como aparece en la siguiente ilustración

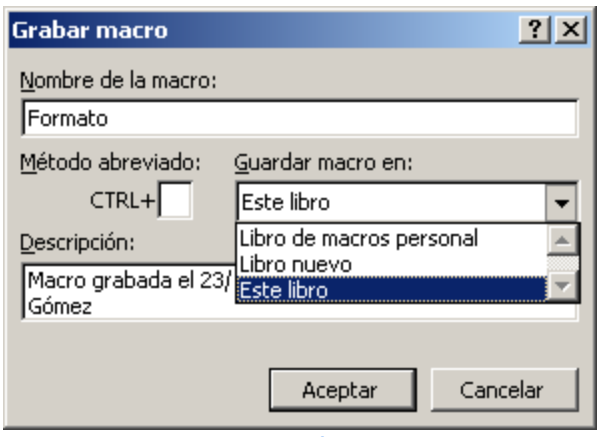

Ilustración 11

La edición de una macro del libro de macros personales se maneja diferente que las de macros particulares de un libro. En este sentido debemos tener en cuenta que este libro de macros personales es un libro oculto y debemos mostrarlo antes de poder editar su contenido. Para mostrar el libro de macros personales y editar una de sus macros debemos realizar los siguientes pasos:

1. En la ventana Excel ir al menú Ventana opción Mostrar con lo que nos aparecerá una ventana similar a la siguiente:

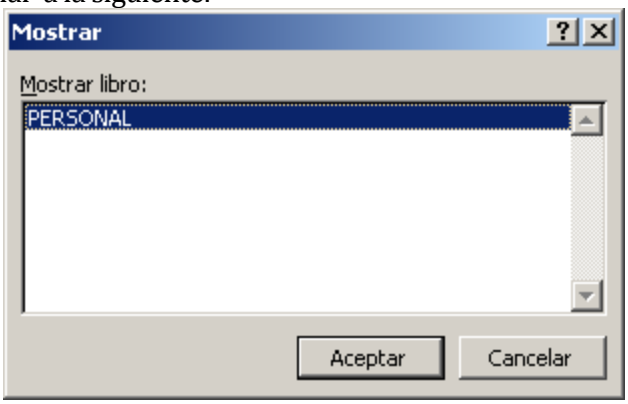

Ilustración 12

2. Seleccionamos Personal y aceptamos con lo que el libro de macros personales se muestra ahora y es el libro activo. Es decir nos aparece un nuveo libro Excel denominado PERSONAL.
3. En este libro PERSONAL vamos a la opciuón Herramientas – Macros y se despliega el cuadro de dialogo (ver Ilustración 4) correspondiente y seleccionando Macros y veremos todas las macros que contiene nuestro libro Personal pudiendo eliminarlas, modificarlas, etc.

2.2 *Eventos*

2.2.1 Introducción.

Hasta ahora todas las macros se ejecutaban cuando se las elige de la lista de macros o se presiona su método abreviado…pero las macros se pueden ejecutar automáticamente…a través de los eventos de la hoja de cálculo y se almacenan en otro contenedor distinto del Módulo.

Recordemos que los Eventos son acciones que ocurren en determinado producido por reacción a determinados estímulos. (Ej. Hacer clic, Hacer Doble Clic, Presionar un botón, Cambiar de Hoja, Introducir un dato en una celda, cambiar de celda, etc.).

2.2.2 Nombre de los Eventos en Excel.

Los Eventos de Excel tienen un nombre definido más complejo y responden a determinado estímulo también definido: **NombreDelObjeto_NombreEvento**.

2.2.3 Eventos de una Hoja Específica de un Libro Excel.

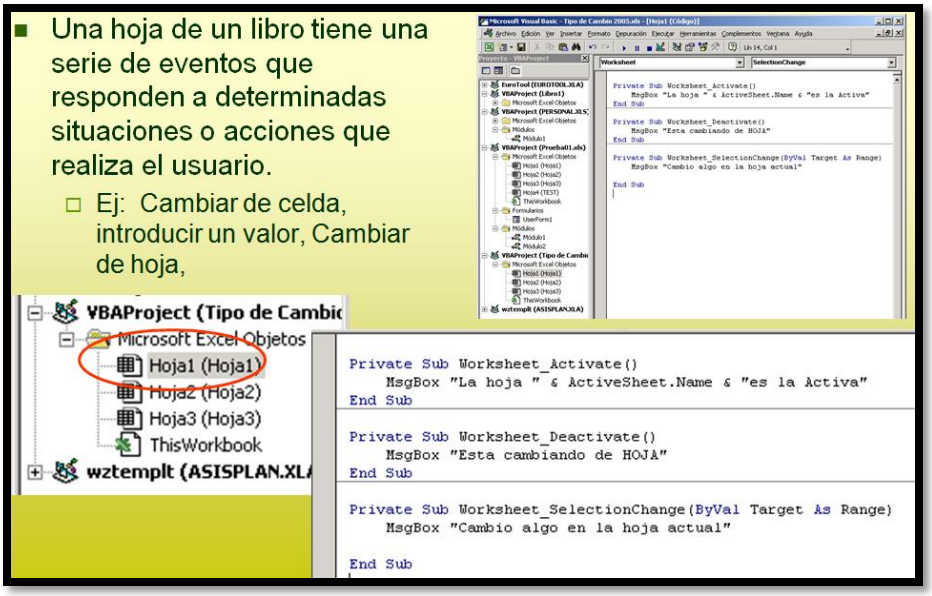

Ilustración 13

Presentamos a continuación una serie de ejemplos de eventos de una hoja de un libro Excel.

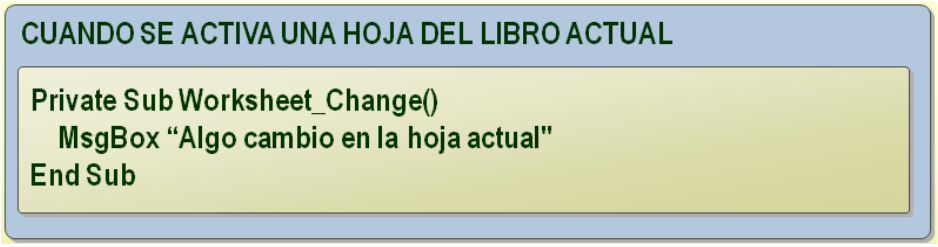

Ilustración 14

2.2.4 Eventos de un Libro Excel.

Un libro tiene más eventos que una hoja de un libro, lo cual permite, tener más alternativas al personalizar Excel.

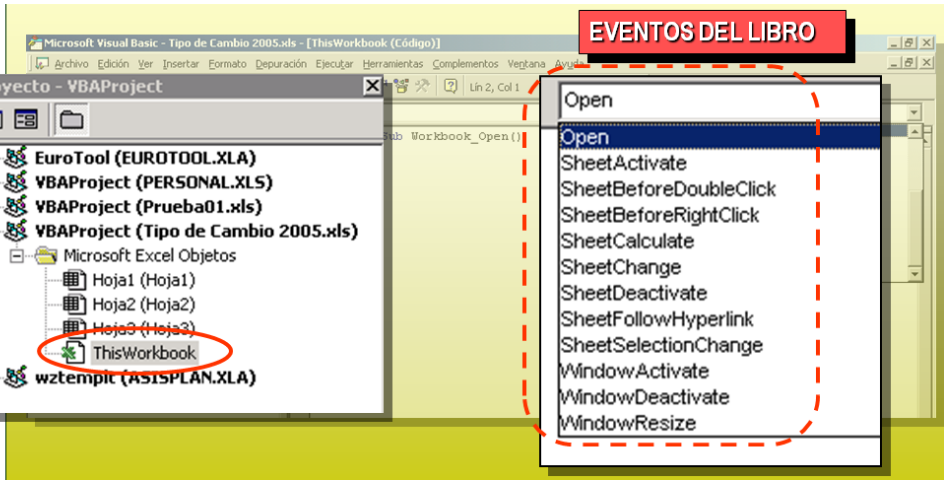

Ilustración 15

2.2.5 Resumen: Objetos, Eventos de libro y eventos de hoja

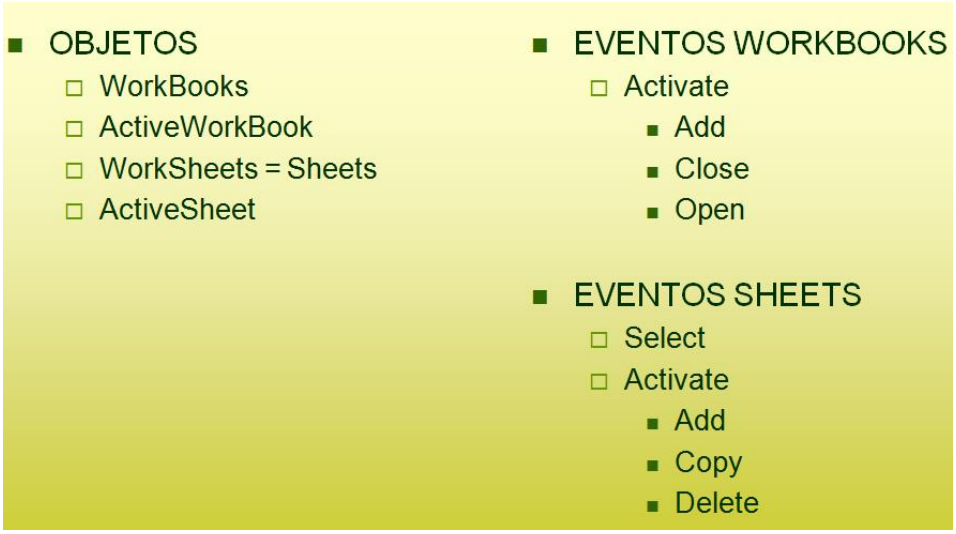

Ilustración 16

2.3 *Ejecutar una macro a través de un botón o vinculada a una celda*

2.3.1 Asignar una macro a un botón

En este caso tenemos creada la macro y simplemente queremos asignar a un botón de formulario o de hoja de tal forma que al hacer doble clic se ejecuta la macro, para ello asignamos al evento doble click que ejecute la macro llamada Restaura_Cuadro_de_Mando, es decir el código quedaría como sigue:

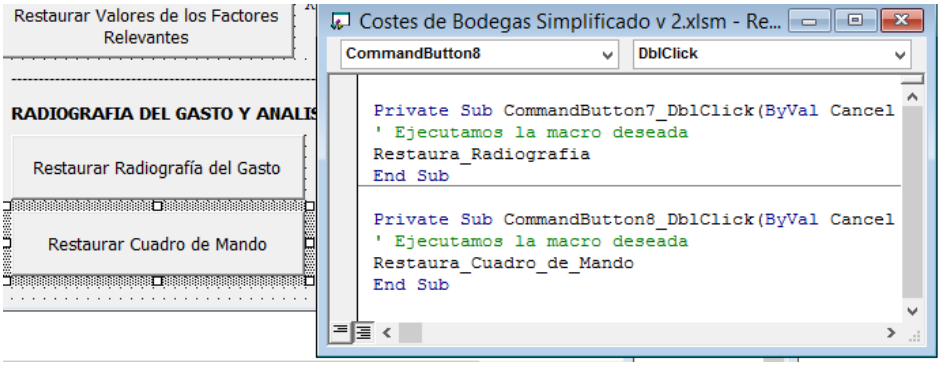

Ilustración 17

2.3.2 Asignar diferentes macros a diferentes celdas

http://www.todoexcel.com/foro-excel/macros/ejecutar-macro-dando-doble-clic-celda-t11044.html
http://www.todoexcel.com/foro-excel/macros/como-ejecutar-macro-con-doble-click-rango-celdas-t16247.html
http://www.todoexcel.com/foro-excel/macros/ejecutar-macro-seleccionar-celda-t13873.html

En este caso queremos que al hacer click o doble click sobre una determinada celda se ejecuta una macro o se realice una determinada acción, en esta ocasión tendríamos el siguiente código,

Por ejemplo queremos que cuando la celda activa o cuando hacemos doble click sobre una determinada celda se ejecuta una acción o macro, en este caso dentro del módulo de la hoja con la que queremos trabajar pondremos el siguiente código que se ejecutara al evento doble click sobre la celda p6

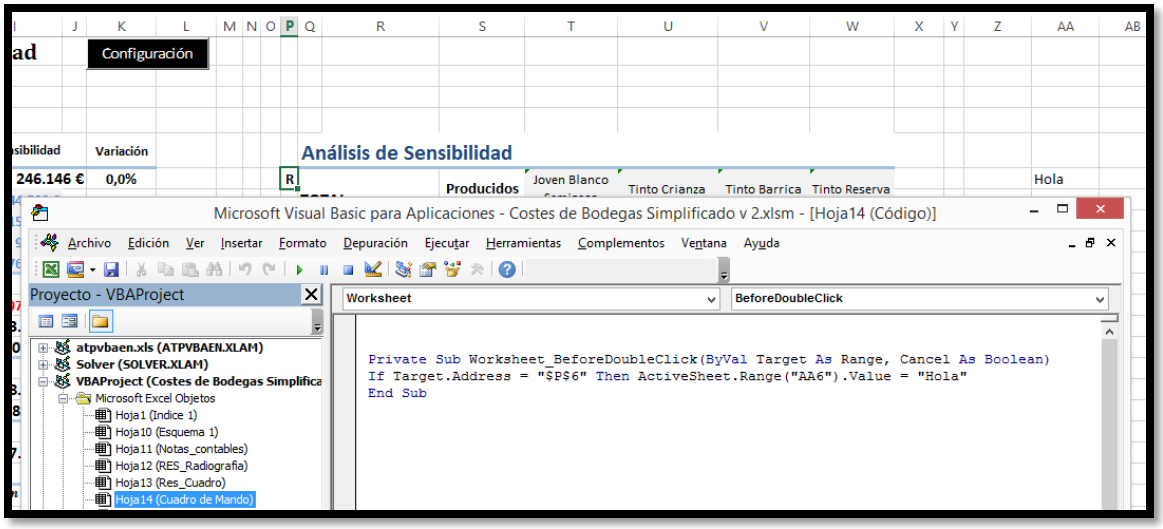

Ilustración 18

Este código tendrá que estar en el módulo correspondiente a la hoja de cálculo

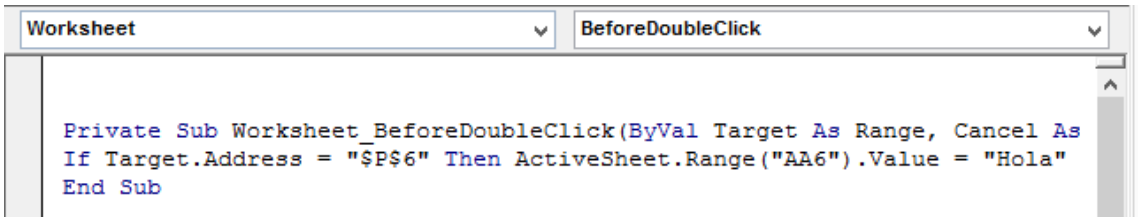

Ilustración 19

Ojo con distinguir mayúsculas y minúsculas

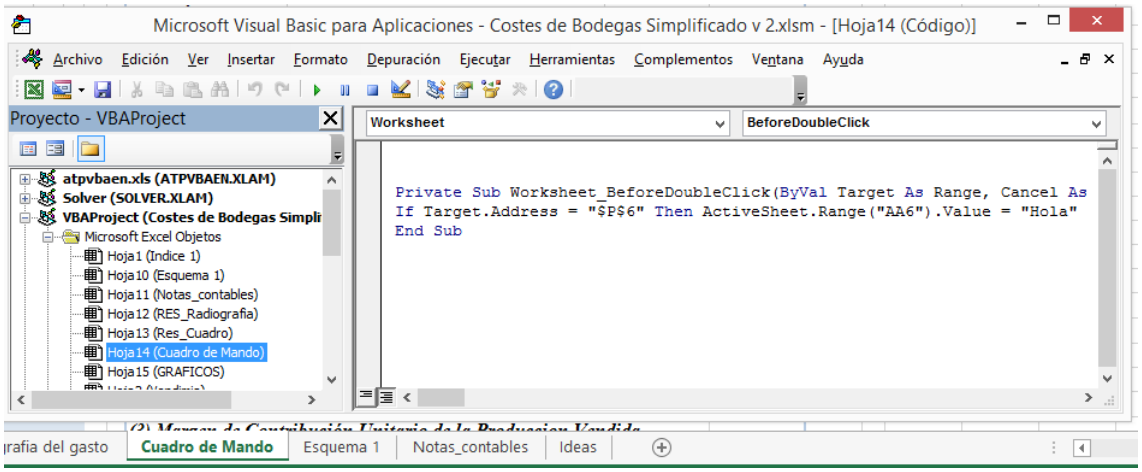

Ilustración 20

En caso de que quisiéramos hacer referencia a un rango de celdas que están combinadas tendríamos que modificar el código anterior por el siguiente:

```
Private Sub Worksheet_BeforeDoubleClick(ByVal Target As Range, Cancel As Boolean)
If Target.Address = "$P$6:$P$8" Then ActiveSheet.Range("AA6").Value = "Hola"
End Sub
```

Ilustración 21

Si quisiéramos en este caso ejecutar o llamar una macro, entonces tendríamos el siguiente código que ejecutaría la macro Restaura_Cuadro_de_Mando

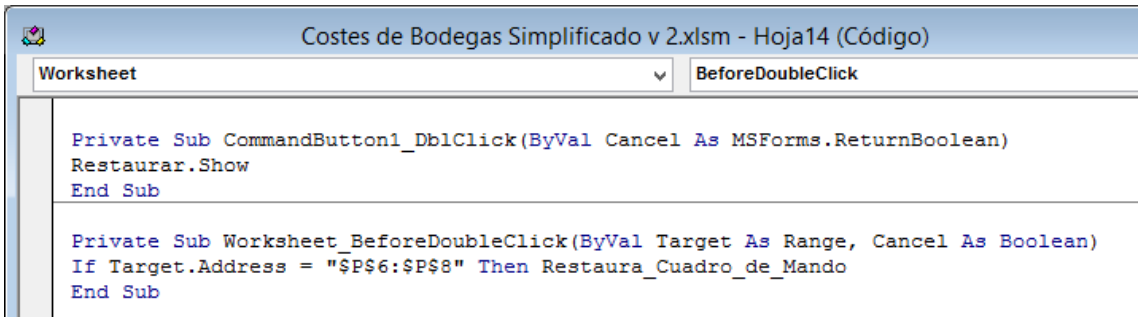

Ilustración 22

2.3.3 Caso especial asignar varias macros a varias celdas al activar o hacer doble click

Si quisiéramos asignar varios macros a varias celdas o rangos de celdas, podríamos adaptar el código anterior para de la siguiente forma, tanto para el evento selección como doble clic

Podrías usar una macro de evento tipo

```
Private Sub Worksheet_SelectionChange(ByVal Target As Range)
If Target.Address = "$B$1" Then Call Macro1
If Target.Address = "$B$2" Then Call Macro2
If Target.Address = "$B$3" Then Call Macro3
etc.
End Sub
```

3 Los módulos

3.1 *Introducción, ¿qué es un módulo?*

¿Realmente es importante saber qué es un módulo? La respuesta es no. Pero es muy útil saber cómo se puede organizar la información dentro de VBA. Pero, ¿por qué es útil?. A la hora de hacer un pequeño programa en Excel con unas cuantas Macros puede ser bueno organizar la información en diferentes módulos.

Un módulo es dónde se escribe el código en VBA. Un proyecto de VBA es un conjunto de módulos (o uno sólo) dónde se escriben las diferentes macros y funciones de VBA. ***Los módulos nos permiten organizar el código de VBA en diferentes archivos dentro de VBA de manera rápida, fácil e intuitiva***.

3.2 *Cómo acceder a un módulo de VBA y como crearlo*

Para acceder a un módulo de VBA primero habrá que acceder al proyecto y después elegir el módulo que se quiere editar.

Para acceder al editor de VBA puedes hacer click en Alt + F11 y accederemos a una pantalla como la siguiente.

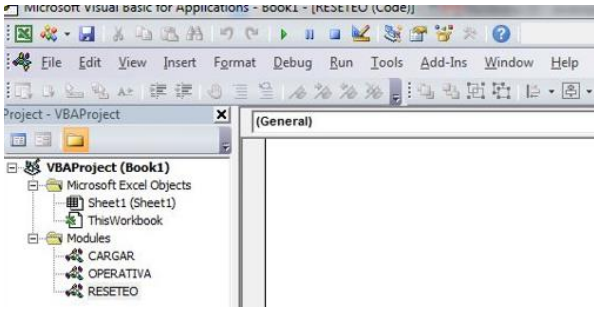

Ilustración 23

En esta pantalla, si hubiera más de un proyecto veríamos un listado a la izquierda de los posibles VBAProjects a elegir. Seleccionando uno de ellos podríamos ver después la carpeta módulo y haciendo click sobre uno de ellos entraríamos a ver el código del módulo seleccionado.

Así para insertar un módulo en Excel, primero debes acceder a la ventana del editor de VBA, como hemos mostrado anteriormente.

1. Una vez aquí hacemos un clic derecho en el Panel de Proyecto, sobre cualquiera de los objetos existentes (el libro y las hojas) y elegimos la opción "Insertar".

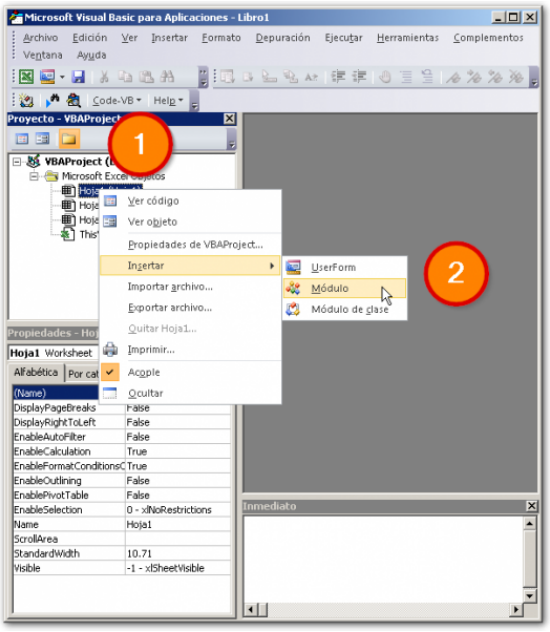

Ilustración 24

2. Ya en el sub-menú que se muestra, elige la opción "Módulo".

¡Y listo! Ya listo ya tenemos un nuevo módulo creado y listo para colocar el código en él.

3.3 *Ejemplo de módulo de VBA*

Pongamos un ejemplo. Tenemos pensado hacer unas cuantas macros para un programita que estamos creando. Podrían ser las siguientes:

1.Abrir un documento
2.Limpiar cierta información
3.Pegar información de una parte del documento a otra.
4.Crear una nueva hoja con cierta información.
5.Aplicar formato a la nueva hoja creada.
6.Eliminar la información copiada.
7.Eliminar la hoja que hemos creado.

Como vemos en el listado anterior, tenemos información de 2 o 3 tipos diferentes relacionadas con la actividad de nuestra hoja de Excel. Las macros de la 1 a la 3 pueden ser la de cargar datos. Las macros 4 y 5 serían de operar con la nueva información. Las macros 6 y 7 son las de dejar la hoja en su estado inicial.

Podria ser útil tener 3 módulos (especie de carpetas donde se guarda la información.

Los módulos yo los organizaría de la siguiente manera:

1.CARGAR
2.OPERATIVA
3.RESETEO

Este es un ejemplo pero depende de la cantidad de macros y la operativa que usemos pueden ser más o menos módulos.

En resumen

> *Un módulo es, en palabras no técnicas, un contenedor para los códigos de tu macros que te ayuda a organizar tu trabajo de una mejor manera. Siempre que tengas porciones de código que no tengan nada que ver con los eventos de las hojas o del libro de trabajo, es mejor que lo coloques dentro de un módulo estándar.*

3.4 *Como llamar a un procedimiento Sub dentro de un módulo. Instrucción Call y Método Run*

El método **Application.Run** y la instrucción **Call** nos servirán para llamar procedimientos o funciones de otro libro de Excel o de un módulo en específico dentro del mismo.

Application.Run (Método)

Este método ejecuta una macro o manda llamar a una función. Se puede usar para ejecutar una macro de Visual Basic o incluso de una función dentro de una DLL.

Call (Instrucción)

Esta instrucción transfiere el control a un procedimiento o una función. Aunque realmente esta instrucción es opcional. Si conocemos el nombre del procedimiento basta con escribirlo en una línea y se ejecuta. Pero es recomendable para cuestiones de lectura de código.

3.4.1 Ejecutar macros de otro libro de Excel con el método Run

Para ejecutar una macro que se encuentre en otro libro de Excel usamos el método Run antes mencionado.

Vamos a suponer que tenemos un Libro2.xlsm el cual contiene un macro llamada Macro1. Usaremos el siguiente código.

Application.Run "Libro2.xlsm!Macro1"

El ejemplo anterior es tomando en cuenta que tenemos en el Libro2 sólo una Macro1. ¿Pero qué sucede si tenemos otra Macro1 en otro módulo? Para eso definimos el nombre del módulo donde está la macro.

Application.Run "Libro2.xlsm!Módulo1.Macro1"

3.4.2 Ejecutar macros de otro módulo del mismo archivo con la instrucción Call

Como vimos en el ejemplo anterior, para llamar una macro del otro módulo, basta con anteponer el nombre de dicho módulo y después el nombre de la macro o función.

Podremos usar la instrucción Call de manera opcional.

Call Módulo2.Macro1

ó

Call ThisWorkbook.Macro1

3.4.3 Ejemplo

Mostramos a continuación un ejemplo ilustrativo de lo anterior. En este caso contamos con un módulo que hemos llamado Mensajes y que contiene diversos procedimientos y rutinas y entre ellos uno llamado Nombre que lo que hace es mostrar un mensaje concreto tal y como se observa en la Ilustración 25.

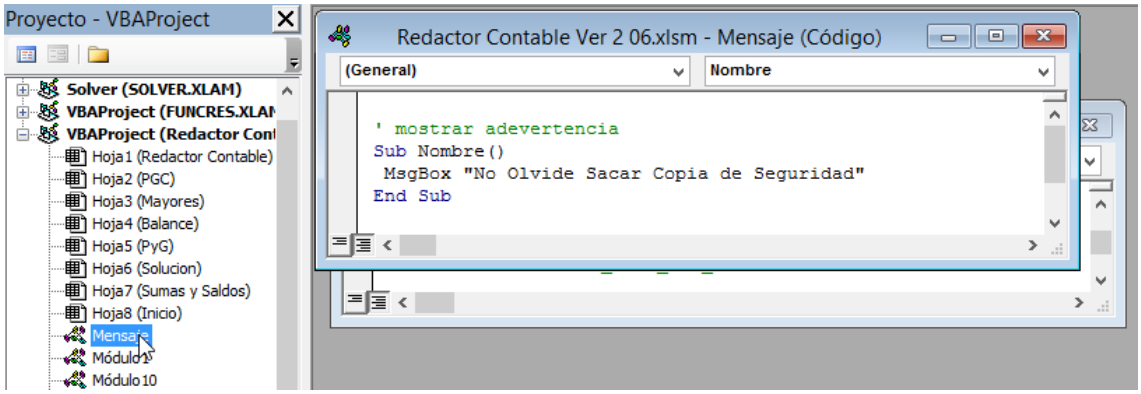

Ilustración 25

En este caso queremos enlazar con este procedimiento a través de un evento concreto por ejemplo al desactivar o cambiar de hoja activa que ejecute el procedimiento Nombre que está en el módulo Mensajes. Para ello tenemos que llamar al citado procedimiento con la instrucción Call haciendo referencia al módulo donde está ubicada tal y como se muestra en la Ilustración 27.

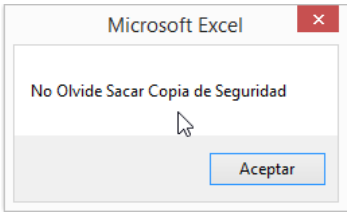

Ilustración 26

Call Mensaje.Nombre

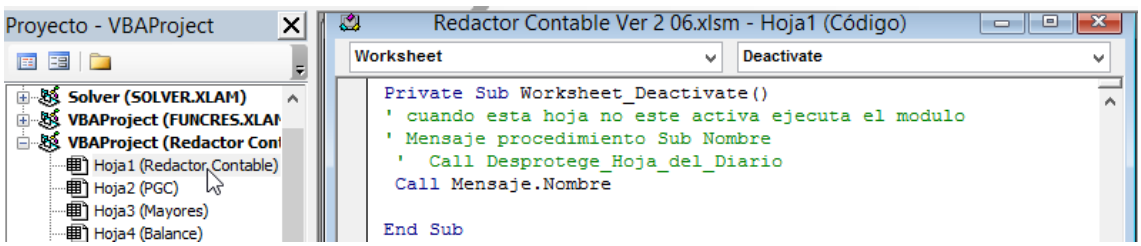

Ilustración 27

4 Bibliografía

http://www.contextures.com/excelvbasendkeys.html
http://excelyvba.com/que-es-un-modulo-de-vba/
http://excelyvba.com/que-son-los-eventos-en-vba/
http://raymundoycaza.com/como-insertar-un-modulo-en-excel/
http://blogs.itpro.es/exceleinfo/2013/10/22/ejecutar-macros-de-otro-archivo-de-excel-con-run-y-call/