Macros en Excel 2003

1.1. Introducción

Este manual fue escrito porque tuve la necesidad de aprender (y aún sigo aprendiendo) a usar Excel, especialmente las macros y su programación aplicadas a la toma de decisiones basadas en el análisis estadístico, este aprendizaje fue impulsado por la necesidad de conseguir empleo como Analista de Datos en México, el conocimiento de Excel es necesario porque que la mayoría de los empleadores exigen como requisito tener un buen dominio de este paquete de computo. Cabe mencionar que no sabía ni macros ni Visual Basic; por lo que podía elegir la opción de no aprender Excel y conseguir otro tipo de empleo, pero la realidad me enseño que están mejor pagadas aquellas personas que lo dominan de los que no (aunque exista software más especializado). Ahora, después de algunas entrevistas y exámenes aplicados por los diferentes departamentos de Recursos Humanos de cada empresa, comprendí esencialmente que es lo que los empleadores buscan en los candidatos, por lo que en los ejemplos de este tutorial presento muchas de las interrogantes que tuve que resolver en la aplicación de esos exámenes laborales, le comento hice bastantes y en cada uno anotaba las preguntas que no sabía responder, de tal forma que cuando llegaba a casa era un reto contestar lo que no pude, y así me preparaba para el siguiente examen que me harían en la siguiente entrevista hasta que por fin domine en menos de un tres mes lo suficiente de macros y su programación para quedarme en mi primer empleo. Estos ejemplos me ayudaron y sinceramente espero que a usted lector le ayuden también, quizás como guía práctica en el aprendizaje de Excel (macros y Visual Basic). Es posible que este manual tenga algunos errores tanto ortográficos como en el código, aunque he procurado evitarlos copiando el código tal y como lo he ejecutado exitosamente, para que disminuyan las erratas, sin embargo, existirán. Le suplico ponga en mi conocimiento, si llega a encontrar, dichos errores al correo macjoerger@hotmail.com. Gracias.

1.2. ¿Por qué estudiar macros?

Es frecuente que algunos usuarios de Excel tengan que hacer tareas repetitivas, estas en ocasiones son operaciones complejas y posiblemente con características propias no definidas en las funciones preinstaladas de Office; pues bien las macros con un poco de conocimiento del lenguaje Visual Basic ayudan en gran medida a disminuir el trabajo automatizando las tareas rutinarias. Aunque es necesaria la programación, no se necesita ser un experto para poder hacer Macros, además, con unos buenos ejemplos y el código bien explicado o documentado lo único que se necesitará es modificarlos a las necesidades requeridas. El lenguaje de código de macros para la mayoría de los programas de Office, incluido Excel, es Visual Basic para Aplicaciones (VBA). Cuando se graban macros en Excel realizamos acciones que el programa guarda automáticamente. Cuando se graba una macro, Excel registra el código de VBA que describe las acciones efectuadas en un módulo asociado al libro. Se escucha mucho pero es necesario aprenderlo, el motivo es disminuir el trabajo cuando se trabajan con muchos datos y los mismos procesos día a día.

En esta primera parte explico que es una macro, como se graba y se expone un ejemplo. Sin embargo, cuando necesitamos tratar a los datos especialmente y no hay herramientas disponibles es útil manipular las macros escribiéndolas con código Visual Basic que más adelante explicaré.

1.3. ¿Qué es una Macro?

Una macro consiste en una serie de comandos y funciones[1] que se almacenan en un módulo[2] (contenedor para almacenar macros) de Microsoft Visual Basic y que puede ejecutarse siempre que sea necesario realizar la tarea. En otras palabras una Macro es un fragmento de código, frecuentemente escrito en Visual Basic, que produce un efecto concreto que lleva su propio nombre, es decir, se le asigna un nombre al fragmento de código (o macro). Existen dos formas de crear una macro: grabarla o escribirla (programarla) en el editor de Visual Basic que esta disponible en la mayoría de las aplicaciones de Office.

1.4. Grabar una macro

Cuando se graba una macro, Excel almacena información sobre cada paso dado cuando se ejecuta una serie de comandos. A continuación, se ejecuta la macro para que repita los comandos. Si se comete algún error mientras se graba la macro, también se graban las correcciones que se realicen, esto ocurre porque se graba todo lo que se hace. Luego Visual Basic almacena cada macro en un nuevo módulo adjunto a un libro.

Recuerde que si comete algún error durante la grabación, no debe preocuparse, porque puede borrar la macro e intentarlo de nuevo.

1.4.1. Ejemplo 1

Este ejemplo ilustra como grabar una macro que permite escribir texto en una celda.

1. Abra Excel y cree un nuevo documento con el nombre "ejercicio01".
2. Elija la celda en la que quiera insertar algún fragmento de texto, por ejemplo su nombre.
3. Seleccione en la barra de Menús <Herramientas/Macro/Grabar nueva macro…>

---

[1] Una función es una fórmula escrita que toma por lo menos un valor, realiza una operación y devuelve como mínimo un valor. Las funciones se utilizan para simplificar y acortar fórmulas (obviamente no escritas) en una hoja de cálculo, especialmente aquellas que llevan a cabo cálculos prolongados o complejos.

[2] Un modulo es una colección de declaraciones, instrucciones y procedimientos almacenados juntos como una unidad con nombre. Existen dos tipos de módulos: módulos estándar y módulos de clase.

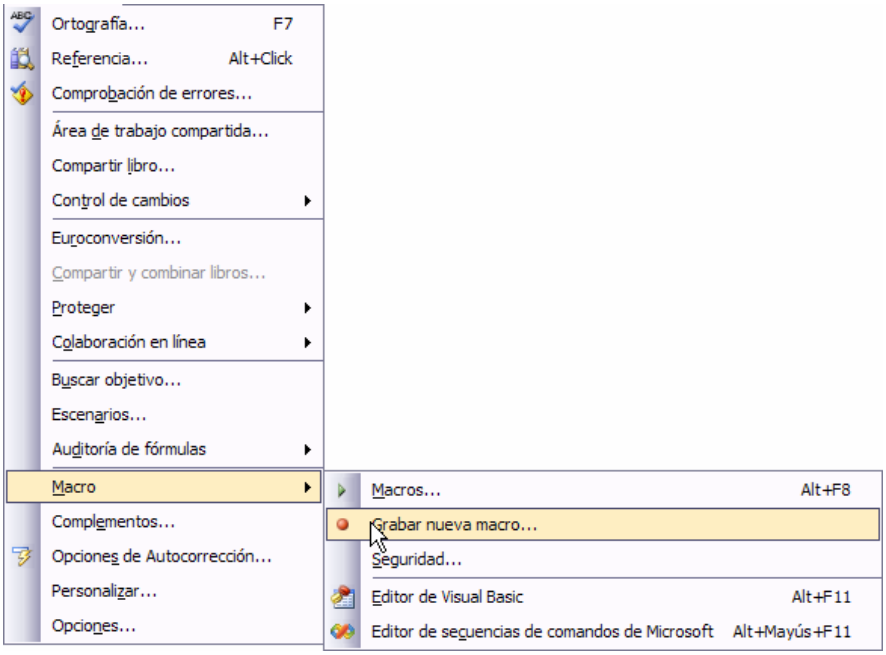


Figura 1

4. En el cuadro de diálogo en la caja de texto de Nombre de la macro[3] escriba: "miPrimerMacro", seguido en el cuadro de texto para Método Abreviado escriba "m".

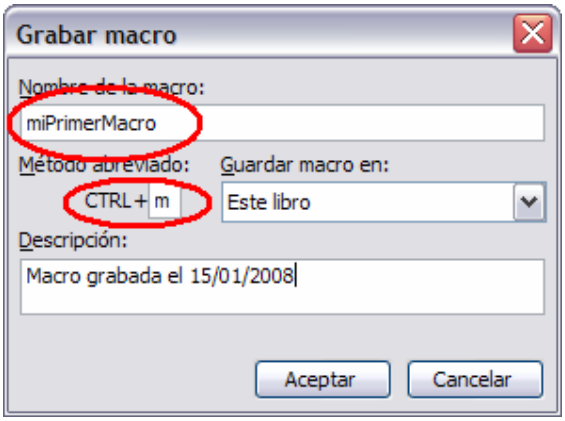


Figura 2

5. Presione el botón aceptar del cuadro de diálogo "Grabar macro". Observe que aparece una barra de herramienta flotante con pocos componentes. A partir de este momento todo lo que haga se grabará en la Macro. Escriba su nombre en la celda seleccionada seguido presione la tecla Enter.

[3] El nombre de la macro puede ser escrito hasta con un máximo de 25 caracteres, necesariamente sin espacios.

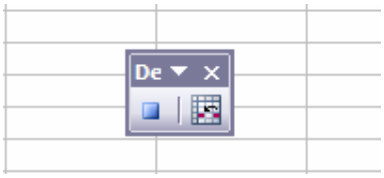


Figura 3

6. Presione el botón detener (el cuadro azul de la Figura 3). Se ha acabado de grabar su primer macro.

1.4.2. Ejecutar una macro

Para ejecutar una macro existen varias formas, la primera es ir a la barra de menú <Herramientas/Macro/Macros…> dar clic en Macros… y aparecerá un cuadro de diálogo como la Figura 4 en seguida seleccione la macro que desea ejecutar y haga clic en el botón <Ejecutar>. La otra forma es, si asignó un método abreviado entonces utilícelo ¿cómo?, en el ejemplo anterior se le ha asignado un método abreviado el cuál activa a la macro, para activarla sólo necesita presionar la tecla "ctrl" y sin dejar de presionarla la letra "m", esta acción la denotaremos en lo siguiente con "ctrl+m"

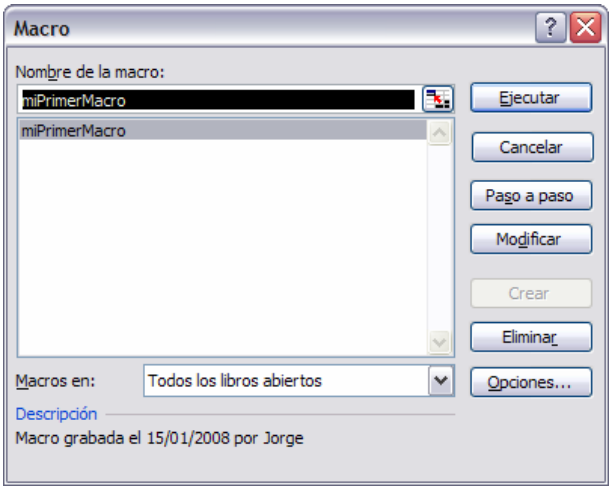


Figura 4

Importante. Si ejecuta la macro y ve que no sucede algún cambio en el libro e hizo bien los pasos anteriores bien, entonces, primero borre el texto de la celda que eligió inicialmente, ejecute la macro y ahora sí, debe insertarse el texto escrito con anterioridad. En caso de que esto no ocurra haga lo siguiente: seleccione <Herramientas/Macro/Macros…> y seleccione la macro con el nombre "miPrimerMacro" seguido presione el botón eliminar y vuelva a hacer los pasos desde el principio.

Escribir macros en el Editor de Visual Basic

1.5. ¿Qué es el editor de Visual Basic?

Lo primero que debemos saber es qué es el Editor de Visual Basic. Entonces el Editor de Visual Basic es un programa diseñado para que los usuarios de Office puedan escribir y editar el código necesario para crear las macros. Además el Editor de Visual Basic permite

copiar las macros de un módulo a otro, entre diferentes libros, cambiar el nombre de los módulos que almacenan las macros o cambiar el nombre de las macros.

1.5.1. ¿Qué es Visual Basic para aplicaciones?

Microsoft Visual Basic para Aplicaciones (VBA) es un lenguaje de programación de alto nivel desarrollado para la creación de aplicaciones Windows. Este cuenta con un conjunto común de instrucciones VBA que pueden ser utilizadas con todos los productos de Microsoft Office, y además cada producto posee su propio conjunto. VBA incluye cientos de órdenes, además de otros controles. Es posible utilizar VBA para integrar características de Word, Excel, etcétera.

1.6. Conceptos básicos

Los **programas** en VBA se denominan **procedimientos** o simplemente **código**. Estos procedimientos también son llamados **módulos**, existen dos tipos de módulos: los **módulos de clases**, que están asociados a un determinado objeto, y los **módulos estándar,** que contienen procedimientos generales que no están asociados a algún objeto.

Procedimientos Sub, son series de sentencias en VBA encerradas por las instrucciones Sub y End Sub que llevan a cabo acciones pero no devuelven algún valor.

Procedimientos Function, están encerradas entre las instrucciones Function y End Function y devuelven un valor.

Cada **procedimiento** es un bloque de código que cumple un determinado propósito. Los **comentarios** son anotaciones que ayudarán a entender el propósito del código al programador o a cualquier persona que lea el código. Las palabras en color azul son **palabras clave** reservadas como parte del lenguaje de programación VBA.

1.7. ¿Cómo escribir una macro en Excel?

Ahora deseamos escribir una macro, no grabarla como en el Ejemplo 1. Para poder realizar nuestro propósito es necesario que este instalado el editor de Visual Basic, esto se hace automáticamente cuando se instala la versión completa de Office 2003, si no lo tiene instalado busque el disco e instale el editor, puede también instalar la ayuda de Visual Basic, es recomendable hacerlo. Para escribir una macro necesitamos hacerlo, ¡No leerlo! Por lo que le recomiendo hacer el ejemplo siguiente.

1.7.1. Ejemplo 2

Empecemos por el principio:

1. Cree un nuevo libro de Excel y guárdelo con el nombre “ejercicio02”.
2. Abra el Editor de Visual Basic. En el menú Herramientas, elija Macro y, a continuación, haga clic en Editor de Visual Basic. Recuerde que el Editor de Visual Basic es una herramienta para escribir y modificar código escrito en VBA, como su nombre indica. Usted visualizará un entorno como lo muestra la figura siguiente:

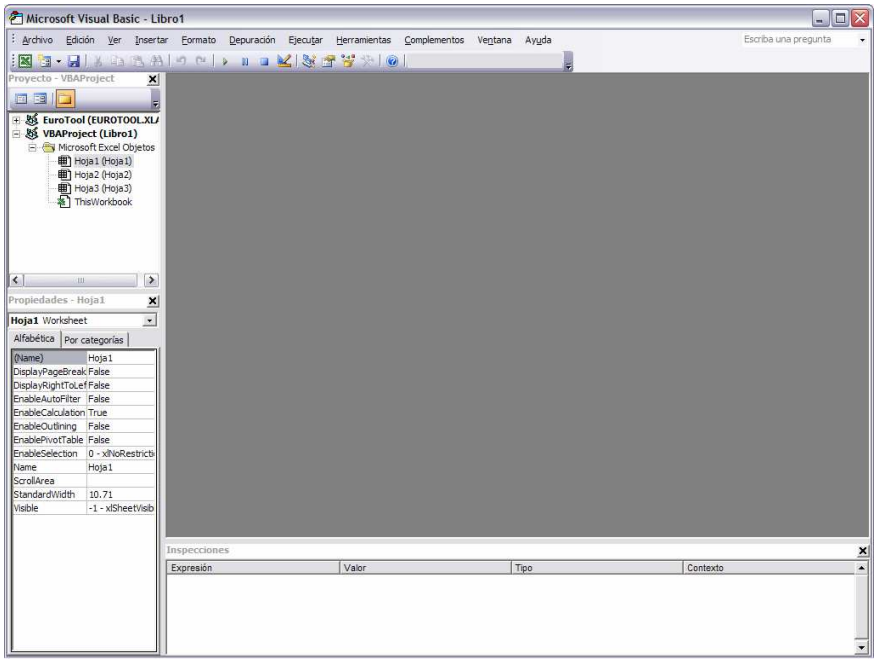


Figura 5

3. Inserte un módulo como un contenedor para almacenar la macro, en el editor de Visual Basic (no en el menú de la hoja de cálculo) vaya al menú <Insertar/Modulo> y haga clic (en Modulo).

4. Escriba la palabra Sub en la ventana del módulo. A continuación, deberá escribir un espacio y luego el nombre de la macro; en este caso "miMacro", seguido presione la tecla Enter. El Editor de Visual Basic insertará automáticamente una línea debajo de la línea Sub con el siguiente texto: End Sub. Este es el principio y el final de la macro; ahora sólo tiene que incluir código en medio, en el nuevo espacio provisto.

```
Sub miMacro()

End Sub
```

5. Entre la línea Sub y la línea End Sub, escriba lo siguiente exactamente:

```
MsgBox "Mi primer macro escrita en Visual Basic"
```

6. Ejecute esta macro desde el Editor ¿cómo?, sólo presione el botón iniciar o ejecutar macro este tiene un símbolo semejante al PLAY de los aparatos electrodomésticos, también puede hacerlo en la barra de menú <Ejecutar/Ejecutar Sub/UserForm> o simplemente presionando la tecla <F5>, Excel mostrará una ventana el texto "*Mi primera macro en Visual Basic*" y un botón **Aceptar** para cerrar el mensaje.

Lenguaje VBA y macros

En la sección anterior se escribió un código sencillo para mostrar un cuadro de dialogo que muestra un mensaje, sólo fue un ejemplo que pretende hacer un acercamiento a la declaración, manejo y ejecución de las Macros. Sin embargo, si queremos resolver problemas o disminuir tiempo en tareas con Excel ya sean escolares, personales o laborales; necesitamos conocer, por decirlo así, más "trucos" o funciones de las macros (estructuras lógicas) para llevar a cabo lo deseado. En esta sección se presentan esos "trucos".

1.8. Usar macros de bucle

En una macro existen bucles (o procesos repetitivos) que funcionan recorriendo los datos y realizando acciones automáticamente repetidas veces. Cuando se trabaja con grandes conjuntos de datos o con datos de distinto tamaño, los bucles pueden ahorrar mucho esfuerzo. Pues bien, a continuación enunciamos los bucles existentes en Visual Basic y su estructura lógica.

1.8.1. El bucle Do…

Este tipo de bucle realiza una acción tantas veces como sea necesario. Contaría el número de filas que encontrara en un conjunto de datos. ¿Cómo sabe el bucle lo que es necesario? Usted se lo indica. El bucle deja de ejecutarse cuando encuentra un segmento de datos específico, como una línea en blanco o un determinado texto. Para especificar cuándo debe parar el bucle Do…, se utiliza la condición **While** o la condición **Until**. El bucle se ejecuta mientras se cumplan determinadas condiciones o hasta que se cumpla una determinada condición. Por tanto, para que un bucle se detenga cuando encuentre una celda en blanco en la primera columna, utilizaría una condición While como la siguiente:

```
Do while Cells(x,1).Value <> ""
```

Aquí se utiliza la condición While para que el bucle se ejecute mientras la celda en la que trabaja no esté en blanco. La celda en la que el bucle trabaja es x, y (x,1) es la primera celda de esa fila, el uno indica que la columna es la primera. Si desea hacerlo en la segunda sólo debe escribir el número 2 por el 1. Utilizados juntos, los signos <> significan "*no es igual a*". Las comillas continuas con nada en medio indican una celda en blanco o vacía. Si quisiera que el bucle se ejecutara hasta que encontrara una celda con el número 30, utilizará la condición Until. De esa forma, le indicaría al bucle cuándo debe parar, y lo hará cuándo encuentre la celda con el número 30.

1.8.2. Ejemplo 3

En este ejemplo se utiliza el bucle Do… para contar el número de celdas con texto en la primera fila, el resultado lo escribirá en otra celda. Siga los siguientes pasos:

1. Abra Excel, cree un nuevo Libro (documento).
2. En la columna A desde el renglón 1 hasta 30 escriba datos, es decir, texto, series de números, etc.
3. Abra el Editor de Visual Basic, <Herramientas/Macro/Editor de Visual Basic>
4. Inserte un nuevo módulo.
5. Escriba el siguiente código dentro del modulo

```
Sub BucleDo()
    x = 1
    Do While Cells(x, 1).Value <> ""
        x = x + 1
    Loop
    Cells(1, 2).Value = x - 1
End Sub
```

6. Desde el Editor de Visual Basic ejecute la macro. Luego revise en el libro que el resultado es el correcto y este escrito en la columna B renglón 1.

1.8.3. Explicación del código.

Cuando escribimos "x = 1", estamos declarando una variable, en este caso "x" y la estamos inicializando con el valor de uno, se le asigna este valor porque las celdas se localizan a través de coordenadas que empiezan en (1,1), el cuerpo Do… esta explicado en los párrafos anteriores, por lo que no creo conveniente repetirlo nuevamente. La sentencia "x=x+1" indica que "x" se incrementará en uno hasta cumplir la condición, de esta forma x contiene el total de iteraciones (repticiones) que hace la setencia Do… en este caso 31 ya que iniciamos desde uno y no desde cero, por ello Cells(1,2).Value = x-1 escribe el valor 30 en la columna B y renglón 1 .

Importante. Si cierra el archivo y lo quiere volver a utilizar la macro, ésta se deshabilitará automáticamente por la seguridad de Excel, tendrá que habilitar las macros. ¿Cómo?, en el menú Herramientas, en Macro, haga clic en seguridad y en el nivel de seguridad elija Medio, ver figura 6. Guarde los cambios y cierre el libro; ábralo de nuevo, aparecerá un cuadro de dialogo llamado "Advertencia de seguridad" haga clic en Habilitar macros, ver figura 7. Ahora si ya puede utilizar la macro. Otra forma de hacer que la macro este disponible es crear un certificado con una firma digital, la cual tenemos que confirmar, pero eso lo veremos después.

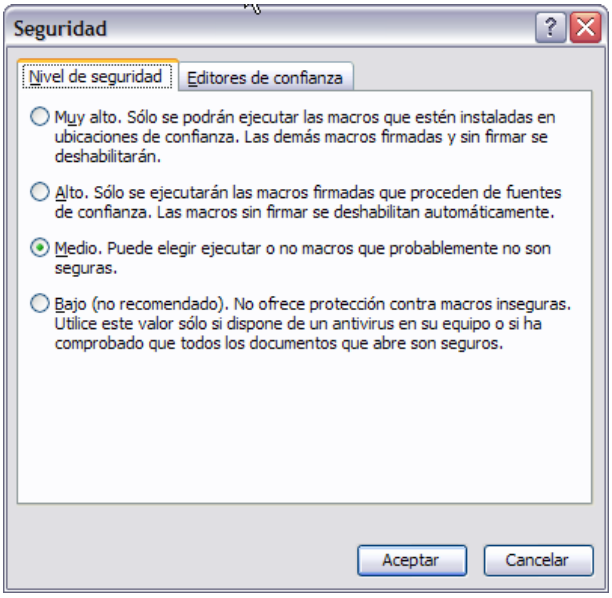
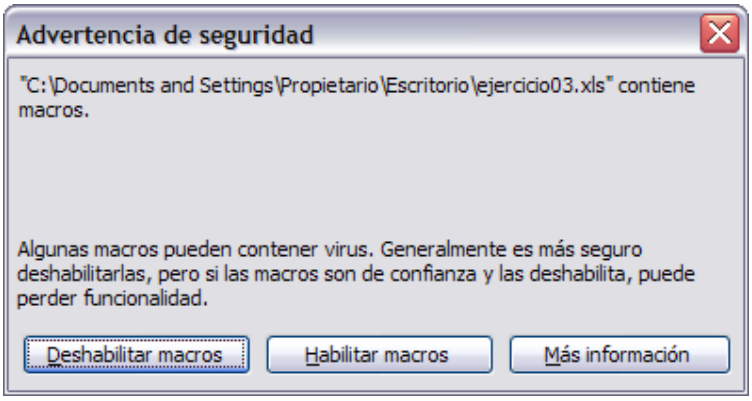


Figura 6



1.9. El bucle For Each…Next

El bucle For Each…Next es semejante al bucle Do… sólo que este repite un grupo de instrucciones para cada elemento de una matriz o colección (grupo). La sintaxis es:

```
For Each elemento In grupo
   [instrucciones]
   [Exit For]
    [instrucciones]
Next [elemento]
```

La parte *elemento* es obligatorio, es una variable que se usa para iterar por los elementos del conjunto o matriz.

El *grupo* es obligatorio, es el nombre de un conjunto de objetos o de una matriz (excepto de una matriz definida por el usuario, programador).

1.9.1. Ejemplo 4

En este ejemplo la palabra "Aceptar" aparecerá más oscura que el resto del texto en cualquier lugar del grupo de datos seleccionado (o celdas seleccionadas). Considero que no es necesario explicar los pasos que se requieren para llegar hasta un módulo por que ya lo he explicado, entonces sólo mostraré de aquí en adelante exclusivamente el código:

```
For Each miCelda In Selection
     if miCelda.Value Like "Aceptar" Then
        miCelda.Font.Bold = True
     End If
Next
```

Para que el código funcione deberá escribir diferentes palabras en cada calda de Excel, en seguida es indispensable seleccionar las celdas a las que se requiere aplicar la macro, seguido necesita ejecutar la macro.

Breve explicación. "miCelda" recorre el conjunto de datos e indica cualquier celda en la que trabaja el bucle, y "For Each" indica que el bucle actuará sobre todas las celdas de la selección. Si el bucle encuentra una celda que contiene la palabra "Aceptar", la mostrará más oscura. (La apariencia del texto se controla mediante la propiedad Font, y la propiedad Bold indica un texto en negrita.).

1.9.2. Ejemplo 5

Este ejemplo cambia de color la celda de una selección que contenga las palabras *libro*, *solo*, y las celdas que estén vacías las dejan del mismo color (blanco) y las otras celdas seleccionadas que no cumplan estas características les cambia al color azul.

```
Sub Color()
   Dim Celda As Range

   For Each Celda In Selection
      If Celda.Value Like "*libro*" Then
         Celda.Interior.ColorIndex = 3
      ElseIf Celda.Value Like "*solo*" Then
         Celda.Interior.ColorIndex = 4
      ElseIf Celda.Value = "" Then
         Celda.Interior.ColorIndex = None
      Else
         Celda.Interior.ColorIndex = 5
      End If
   Next
End Sub
```

Por primera vez esta escrita la palabra *Dim*, sirve para declarar variables, pero se debe de especificar el tipo para ello se escribe la palabra *As* seguida del tipo de dato (entero, real, etc.), en este caso es de rango (Range).

"Celda" es una variable que controla la celda en la que actúa el bucle. Los asteriscos del ejemplo de código permiten que el código busque el texto especificado cuando está integrado en otro texto.

1.10. Bucles anidados

Los bucles Do… y For Each…Next son eficaces aunque simples. Ahora vamos a complicar un poco las cosas introduciendo bucles anidados. Los bucles anidados se utilizan cuando hay que realizar una acción en un grupo de datos varias veces, o a través de varios grupos de datos. Para establecer una analogía con los bucles anidados, considere el movimiento de traslación de la Tierra. Una vuelta completa alrededor del Sol, un año, es similar al bucle exterior, y el movimiento de rotación de la Tierra en torno a su eje es similar al bucle interior anidado dentro del bucle exterior. Cada año se producen 365 bucles interiores, y el bucle exterior se repite cada primero de enero:

```
Do While (la tierra gira alrededor del sol)
   Do While (la tierra gira alrededor de su eje)
      if (frente al sol)
        Día
      else
        Noche
      end if
   Loop
Loop
```

Este código no se ejecutaría en Excel, pero ilustra el hecho de que para cada gran bucle (alrededor del Sol) hay 365 bucles más pequeños (alrededor del eje de la Tierra).

1.10.1. Ejemplo 6

Este ejemplo utiliza un bucle anidado que elimina dentro de una fila los datos duplicados. Esta macro eliminará los datos de su hoja de cálculo. Y cuando se ejecuta una macro, no hay un comando Deshacer.

```
Sub BorraDatosRepetidos()
   x = ActiveCell.Row
   z = ActiveCell.Column
   y = x + 1
   Do While Cells(x, z).Value <> ""
      Do While Cells(y, z) <> ""
         If (Cells(x, z).Value = Cells(y, z).Value) Then
               Cells(y, z).EntireRow.Delete
         Else
            y = y + 1
         End If
      Loop
      x = x + 1
      y = x + 1
   Loop
End Sub
```

El bucle interno busca en toda las fila que datos están duplicado después de seleccionar la celda busca hacia abajo. El bucle externo desplaza la fila inicial hacia abajo de la selección, fila por fila, hasta que compara cada fila con todas las que están debajo en la selección.

¿Cómo utilizar esta macro?

Es necesario tener datos en las columnas del libro de Excel, seleccione la celda a partir de la cual quiere borrar los datos, a partir de esta la macro buscará los datos repetidos y los borrará. Vuelvo a recordar que después de ejecutar la macro no podrán recuperarse los datos. ¡Tenga cuidado!

1.11. Utilizar la propiedad Cells con bucles

Las macros de bucle utilizan dos métodos diferentes para llevar los datos de una celda a su código. Uno es la propiedad Cells y el otro es la propiedad Range. En VBA, suele ser más fácil y cómodo utilizar la propiedad Cells, porque es más sencillo cambiar los valores descritos por esa propiedad. La propiedad Range identifica las filas y columnas mediante los números y las letras de la hoja de cálculo, mientras que la propiedad Cells utiliza números para las filas y las columnas. Agregar +1 a esos números es una forma cómoda de hacer que la macro de bucle se mueva de fila en fila y de columna en columna, pero no hay ninguna forma sencilla de permitir que la macro pase de una letra a la siguiente en el código.

Sugerencia. Si lo prefiere, puede hacer que aparezcan números de columna en lugar de letras en la hoja de cálculo. En el menú Herramientas, haga clic en Opciones y, a continuación, seleccione la ficha General. Active la casilla "Estilo de referencia F1C1". Si desea volver a cambiarlo más adelante, desactive esa casilla de verificación.

1.12. Option Compare (instrucción)

Sintaxis

```
Option Compare {Binary | Text | Database}
```

Si se usa, la instrucción **Option Compare** debe aparecer en un módulo antes de cualquier procedimiento.

La instrucción Option Compare especifica el método de comparación de cadena (Binary, Text o Database) para un módulo. Si un módulo no incluye una instrucción Option Compare, el método de comparación de texto predeterminado es Binary.

1.12.1. Option Compare Binary

Compara cadenas usando como criterio de comparación un orden derivado de las representaciones internas binarias de los caracteres. En Microsoft Windows, la ordenación se determina de acuerdo con la página de códigos. En el ejemplo siguiente se muestra un criterio de ordenación binario típico:

```
A < B < E < Z < a < b < e < z
```

1.12.2. Option Compare Text

Compara cadenas usando como criterio una ordenación de texto que no distingue mayúsculas de minúsculas determinado por la configuración regional del sistema. Cuando se ordenan los mismos caracteres mediante Option Compare Text, se produce la siguiente ordenación de texto:

```
(A=a) < (B=b) < (E=e) < (Z=z)
```

1.12.3. Option Compare Database

Solamente puede utilizar Option Compare Database dentro de Microsoft Access. Esto da como resultado comparaciones de cadena basadas en el orden, que está determinado por el identificador local de la base de datos en la que tienen lugar las comparaciones de cadena.