Ing. Emanuel Bombina

Macros – Programación Orientada a Objetos

En este documento se explicarán algunos conceptos como:

- ✓ **¿Qué es una macro?**
- ✓ **¿Cómo elaborar una macro?**
- ✓ **Programación Orientada a Objetos (POO)**
- ✓ **tres elementos a la hora de programar**
  - **OBJETOS**
  - **PROPIEDADES**
  - **MÉTODOS**
- ✓ **Objetos de objetos**
- ✓ **Variables**
- ✓ **Constantes**
- ✓ **Procedimientos**
- ✓ **Argumentos**
- ✓ **Recuerde los problemas de seguridad**
- ✓ **Antes de empezar, la ayuda**
- ✓ **Nuestra primer macro programada**

¿Qué es una macro?

Una macro son un conjunto de instrucciones que sirven para automatizar procesos. Refiriéndonos a Excel, supongamos que realizamos frecuentemente la acción de seleccionar un rango de celdas para aplicarle negrita, formato de tipo moneda y centrado. En lugar de hacer estas acciones manualmente, se puede elaborar una macro e invocarla para que ejecute los tres procesos automáticamente.

¿Cómo elaborar una macro?

Hay dos formas de elaborar Macros, en ambas deben agregar la pestaña o ficha "**Programador**" o "**Desarrollador**":

1. Grabando Macros, con la opción "**Grabar Macro**".
    - La ventaja es que es mucho más rápido y sencillo.
    - La desventaja es que no es flexible para agregar nueva funcionalidad.

2. **Programando Macros**, trabajando desde el editor VBA.
    - La ventaja es que es mucho flexible para agregar funcionalidad.
    - La desventaja es que se necesitan conocimientos de Programación en VBA.
    - Es la forma más profesional de trabajar con Macros.

Programación orientada a objetos (POO)

Para entender y trabajar mejor con las Macros programadas, hay que entender el concepto de la programación orientada a objetos. POO es una forma de programación en computadoras que surge en los años 70 pero tiene un desarrollo sorprendente en los años 90 al ser utilizada en las microcomputadoras. Se diferencia de la programación clásica o estructurada en que las instrucciones hacen referencia a los elementos del entorno. Esos elementos representan "objetos"; y todos los datos y todas las acciones que se hagan con ellos o sobre ellos, están encapsuladas u ocultas en el objeto.

Un objeto es una entidad provista de un conjunto de propiedades o atributos (datos), de un comportamiento o funcionalidad (métodos) y de sus posibles relaciones con otros objetos.

El concepto de objeto tiene un concepto equivalente al objeto de nuestro mundo real. En nuestro entorno siempre estamos en constante relación con objetos: los creamos, los usamos, los modificamos cambiando sus atributos, características o propiedades, los relacionamos con otros objetos, etc. Por ejemplo tomemos el objeto moto. Una moto es un objeto bastante pesado que tiene un conjunto de propiedades como su patente, color, marca, modelo, accesorios, etc. Tiene también un conjunto de funciones como la de desplazarse, detenerse, ponerse en marcha. Podemos cambiarle de color, aumentar o quitar sus accesorios; es decir, podemos modificar sus propiedades. También podemos mediante un conjunto de métodos darle uso al objeto moto.

Tres elementos a la hora de programar

Objetos

Los objetos son las entidades con las que trabajamos. En el Excel podemos hablar del objeto Celda. Este objeto tiene dimensiones, color de fondo, tipo de borde, tiene un contenido o valor. Posee algunas funcionalidades que nos permiten cambiarlo de tamaño, de color de fondo, de contenido. El objeto celda pertenece al objeto Rango y está relacionada con él y tiene otras relaciones con otros objetos como el objeto Hoja, el objeto Gráfico, Libro, etc.

Propiedades

Las propiedades son las Variables que describen algunos aspectos o características del objeto en el que están incluidas. Las propiedades de un objeto toman un valor que puede ser permanente o puede cambiar. Por ejemplo la propiedad color del objeto coche tomará un valor en concreto: verde, rojo, etc. El valor concreto de una propiedad de un objeto se llama estado del objeto. Podemos modificar la propiedad de un objeto accediendo a su estado. Las propiedades de un objeto pueden tomar uno o varios valores. Estos valores pueden ser de cualquier tipo de dato (String o cadena de caracteres; entero, etc.).

Para acceder al estado de un objeto en POO se usa la siguiente sintaxis:

MiMoto.Color = Azul

Dónde el punto recibe el nombre de operador.

Aquí, MiMoto es una instancia del objeto Motocicleta; vale decir, es una copia.

Una propiedad de muchos objetos en Excel es Nombre. El objeto celda, rango u hoja tiene un nombre cuyo valor es asignado por defecto por el Excel o es asignado por el usuario. Una forma de acceder a la propiedad Nombre del objeto rango será:

ActiveSheet.Name = "Ingresos"

En este caso el objeto Hoja activa está cambiando de nombre por Ingresos.

Métodos

Un método es una acción que el objeto "reconoce" y "sabe" cómo ejecutarlo. Es una acción u operación que realiza acceso a los datos. Se puede definir como un programa o procedimiento escrito en algún lenguaje que está asociado a un objeto determinado y cuya ejecución sólo puede desencadenarse a través de un mensaje recibido por el objeto o por sus descendientes. El objeto moto reconoce al procedimiento "Frenar" y sabe cómo debe realizar la acción de frenado. Del mismo modo, en Excel, el objeto Hoja puede ser declarada como activa. El método que permite activar la hoja "Ingresos" es

```
Sheets("Ingresos").Select
```

La hoja reconoce este método y dicha hoja pasa a ser activa ubicándose en primer plano y con mayores prioridades que las otras hojas.

Ejemplo de objeto, propiedades y métodos:

Objeto: Alumno

El objeto alumno tiene un conjunto de atributos o propiedades como: Edad, sexo, peso, altura, nombre, raza, color de cabello, etc.

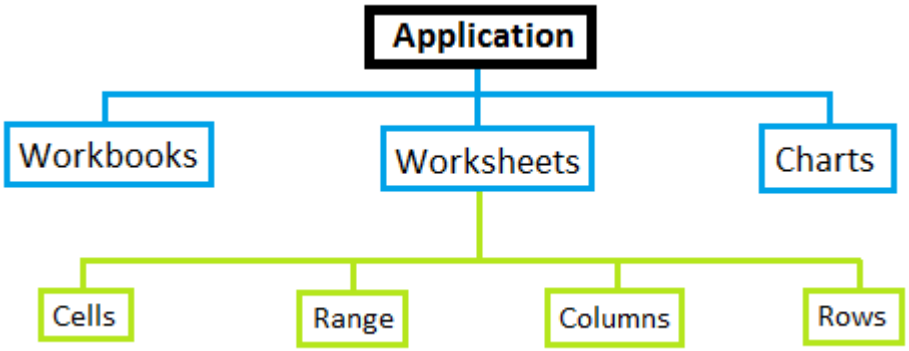
Existen un conjunto de acciones o métodos, que se realizan sobre él: hablar, comer, dormir, caminar, vestirse, correr, detenerse, etc.

Algunas de estas propiedades son heredadas de sus padres, otros objetos de jerarquía superior.

El objeto alumno está relacionado con otros objetos como hermano, amigo, vecino. Varios de estos objetos forman una clase: la clase Persona.

Objetos de objetos

Es muy habitual que una propiedad de un objeto sea otro objeto. En Excel, el objeto **WorkSheets** tiene la propiedad **Range** que es un objeto, **Range** tiene la propiedad **Font** que es también un objeto y **Font** tiene la propiedad **Bold** (negrita). Se utilizará en forma frecuente Propiedades de un objeto que serán también Objetos. Dicho de otra forma, hay propiedades que devuelven objetos**..**



Variables

A continuación dejaremos que el usuario entre un texto desde teclado y a continuación guardaremos ese valor en la celda A1. Observe que el valor que entre el usuario debe guardarse en algún lugar para poder ponerlo después en la celda A1; pues bien, ese valor se guardará en una variable. Una variable es simplemente un trozo de memoria que la macro se reserva para guardar datos, la forma general de declarar una variable es

DIM variable AS tipo

Siendo variable el nombre que se asigna a la misma y Tipo el tipo de datos que se guardarán (números, texto, fecha, boleanos, etc.) En nuestro ejemplo, declararemos la variable de tipo String (tipo texto), y lo haremos de la forma siguiente.

Dim Texto As String

Con esto estamos indicando que se reserve un trozo de memoria (el que sea) , que se llama Texto y que el tipo de datos que se guardarán ahí serán caracteres.

Las variables pueden ser: **Locales**, **Públicas** o **Estáticas:**

- Las **variables Locales** son aquellas que se declaran dentro de un módulo o procedimiento y sólo pueden ser utilizadas en éste. Éstas dejan de existir una vez que el procedimiento termina su ejecución. Para declararlas se debe usar la sentencia DIM.

- Si se quiere que una variable esté disponible para todos los procedimientos de todos los módulos VBA de un proyecto, se la debe definir a través de la sentencia PUBLIC (y no DIM). Por ejemplo: **Public NTotal As Integer**. Las variables públicas se deben definir antes del primer procedimiento de un módulo de VBA.

- Si se desea que una variable definida en un procedimiento conserve su valor una vez terminado éste, e ingresado a otro procedimiento, ésta se debe definir a través de la sentencia STATIC. Por ejemplo **Static nDat As Integer.**

Constantes

A diferencia de la variables, cuyo valor cambia al ejecutarse un procedimiento, hay valores que no cambian durante la ejecución de un procedimiento, éstos valores se denominan Constantes. Las constantes se definen a través de la sentencia Const. Por ejemplo: **Const Nivel As Integer**

Las constantes también pueden declararse como Públicas para que estén disponibles en todos los procedimientos de todos los módulos, esto se hace a través de la sentencia PUBLIC

Public Const TasaActiva As Integer Esta sentencia debe incluirse en un módulo antes del primer procedimiento. Para definir constantes Locales, basta definirlas a través de la sentencia Const dentro de un procedimiento o función.

Procedimientos

Un procedimiento está formado por un conjunto de sentencias que permite resolver un problema. Un módulo, que es un ambiente de trabajo, está formado por uno o más procedimientos. Un procedimiento se declara a través de la sentencia Sub y puede ser Privado o Público.

Un **procedimiento privado** sólo es accesible por otros procedimientos dentro del mismo módulo. Su sintaxis es:

```
Private Sub Procedimiento (Argumento1,Argumento2,.....)

    [sentencias]

End Sub
```

Un **procedimiento público** es accesible por todos los procedimientos de todos los módulos VBA de un proyecto, su sintaxis es:

```
Public Sub Procedimiento(Argumento1,Argumento2,.....)

    [Sentencias]

End Sub
```

La sentencia Sub y End Sub son obligatorias al definir cualquier procedimiento. Los argumentos y las sentencias Private y Public son opcionales. Es importante mencionar que al definir un procedimiento sin ninguna de las sentencias anteriores, por defecto éste se define como Público.

Para llamar a un procedimiento desde otro procedimiento, se puede utilizar la sentencia Call o simplemente el nombre del procedimiento. Por ejemplo:

```
Sub Proced1 (Argumento1,Argumento2,.....)

    [Sentencias]

    Proced2

    [Sentencias]

End Sub
```

En este caso, el procedimiento Proced1 llama al procedimiento Proced2.

La sentencia Call se utiliza cuando se requiere llamar a un procedimiento al cual hay que pasarle un argumento.

Argumentos

Los argumentos pueden ser pasados a un procedimiento por referencia (por defecto los argumentos se pasan de esta forma) o por valor. Cuando un argumento es pasado por referencia, se pasa la variable misma al procedimiento llamado, por lo que los cambios que se producen en la variable son devueltos al procedimiento principal (al que llamó al otro). En cambio cuando un argumento es pasado por valor, se pasa una copia de la variable al procedimiento llamado por lo que los cambios que se producen en la variable no son devueltos al procedimiento principal. Para pasar un argumento por valor, se utiliza la sentencia ByVal, por ejemplo:

```
Sub Proced2( ByVal indice)

    [Sentencias]

End Sub
```

Pasar argumentos por valor es útil cuando se requiere conservar el valor original de una variable después de llamar a otro procedimiento.

Al especificar los argumentos de un procedimiento también es posible definir el tipo de dato, por ejemplo se puede definir un procedimiento de la siguiente forma:

```
Sub Procedimiento(argumento1 As Integer, argumento2 As String)

    [Sentencias]

End Sub
```

Recuerde los Problemas de seguridad

Haga clic en el botón de **Seguridad de macros** para especificar qué macros pueden ejecutarse y en qué condiciones. Aunque el código de macros de sistemas no confiables puede dañar gravemente el equipo, las condiciones de seguridad que impiden ejecutar macros útiles pueden disminuir en gran medida la productividad. Para el propósito de este curso, tenga en cuenta que si la barra **Advertencia de seguridad: las macros se han deshabilitado** aparece entre la cinta de opciones y el libro, cuando abre un libro que contiene una macro, puede hacer clic en el botón **Habilitar contenido** para habilitar las macros. Y recuerde que como medida de seguridad, no puede guardar una macro en el formato de archivo predeterminado de Excel (.xlsx); debe guardar la macro en un archivo con extensión especial (.xlsm).

Antes de empezar, la ayuda

Para ver la ayuda presionamos la tecla F1 en Excel, recuerde hacer búsquedas por palabras clave, en caso de no poder visualizar nada haga clic aquí**.**

Nuestra primer Macro programada

Para crear una primer macro hacemos clic en la pestaña “Desarrollador”, luego el comando “Visual Basic” o combine las teclas ALT+F11. Corrobore estar visualizando el **Explorador de proyectos,** la **Ventana de propiedades** y la ventana de **Código (F7)**. Éstos podemos habilitarnos del menú **Ver** en caso que no estén visibles.

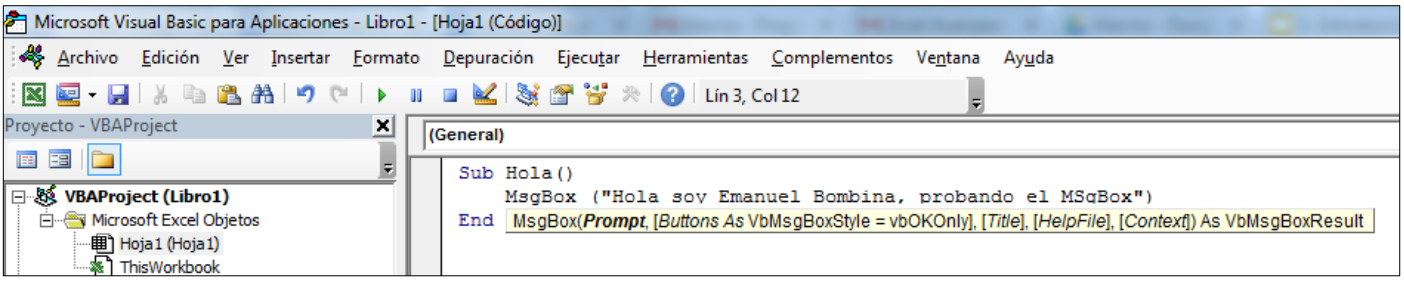
En el código escribimos:

```
Sub Hola()

End Sub
```

Sub indica el inicio de la macro Hola.

La función MsgBox tiene parámetros opcionales para configurar:

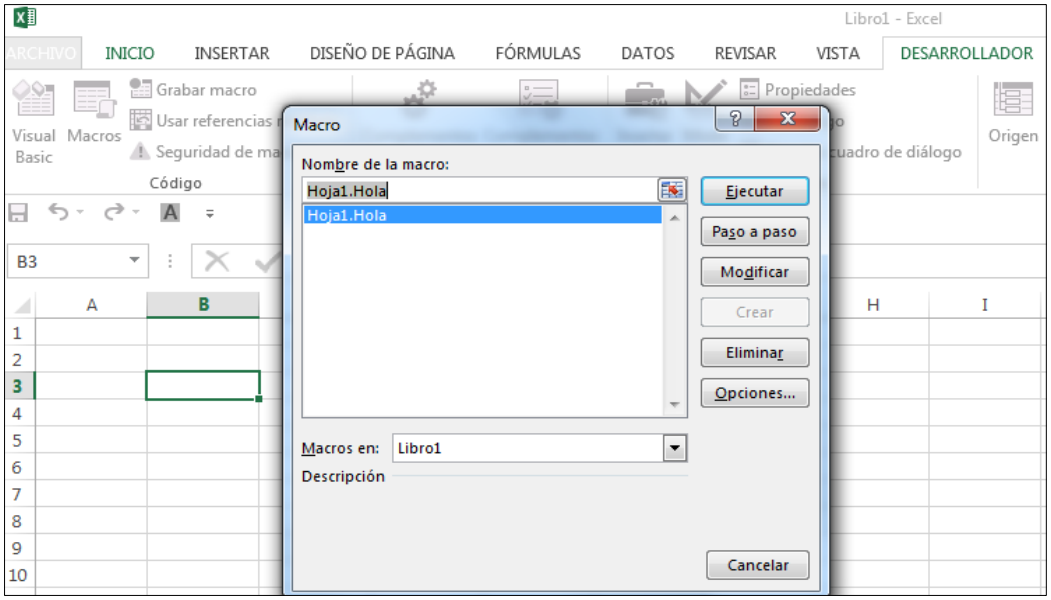


Entonces la Macro es:

```
Sub Hola()
   MsgBox ("Hola soy Emanuel Bombina, probando el MSgBox")
End Sub
```

Para ejecutarla desde Excel:

1. Clic en “Programador” o “Desarrolladora”
2. Clic en “Macros”
3. Seleccionar la Macro “Hola”
4. Presionar el botón de “Ejecutar”



5. Luego de seleccionarla y presionar el botón ejecutar observamos la siguiente ventana: