

Taller: EXCEL

Evaluación de funciones y creación de macros con Excel

Duración: 5hrs.

Ing. Imelda Escamilla Bouchán

CONTENIDO

Tema	Página
1.1 Introducción	2
1.2 Objetivo	2
1.3 Evaluación de funciones	2
1.3.1 Funciones definidas por el usuario	2
1.3.2 Errores comunes	3
1.3.3 Evaluando una función con varios tipos de parámetro	4
1.3.3.1 Evaluación con argumentos variables	4
1.3.3.2 Evaluación con argumentos variables y/o constantes	5
1.3.3.3 Constituyendo rangos con un incremento fijo	7
1.4 Creación de macros empleando VBA	8
1.4.1 ¿Qué es una macro?	8
1.4.2 Objetos, propiedades y métodos	8
1.4.3 Conceptos útiles para trabajar con macros en Excel	9
1.4.4 Grabar una macro	10
1.4.4.1 Mi primera macro	10
1.4.4.1 Ejecutar una macro	13
Práctica 1. “Grabar macros”	14
1.4.5 Manipulación de macros	15
1.4.6 Editor de Visual Basic	16
Práctica 2. “Observando los códigos de una macro en Excel”	20
1.4.7 Cuadro de control	22
Práctica 3. “Formulario”	32
Bibliografía	33

1.1 Introducción

Microsoft Excel © es un software para el manejo de hojas electrónicas agrupadas en *libros* para cálculos de casi cualquier índole. Entre muchas otras aplicaciones, es utilizado en el tratamiento estadístico de datos, así como para la presentación grafica de los mismos. La hoja electrónica Excel es ampliamente conocida, en forma generalizada, por profesionales y estudiantes en proceso de formación, pero hay una gran cantidad de usuarios que no conocen a profundidad su gran potencial y adaptabilidad a los diferentes campos del conocimiento.

1.2 Objetivo


El siguiente documento tiene como objetivo explicar cómo evaluar funciones y realizar macros en Excel en el entorno gráfico, así como algunos ejemplos mediante el lenguaje Visual Basic for Applications (VBA)¹.

1.3 Evaluación de funciones

1.3.1 Funciones definidas por el usuario

Para ejemplificar este tópico, supongamos la siguiente función a evaluar

$$f(x) = 2x^3 + \ln(x) - \frac{\cos(x)}{e^x} + \sin(x)$$

1. Como al evaluar $f(x)$ se debe recurrir a varias funciones básicas predeterminadas de Excel, se puede tener acceso a su sintaxis, pulsando el ícono  y seleccionar “Matemáticas y Trigonómicas”.
2. Para escribir esta fórmula, primero ponemos un valor para x en la celda B3.
3. Ahora en la celda C3 introducimos la fórmula (de acuerdo a la sintaxis de la versión de Excel en español):

$$=2*B3^3+LN(B3)-COS(B3)/EXP(B3)+SENO(B3)$$

Una vez introducida, simplemente se pulsa la tecla “Enter” Figura 1.1.

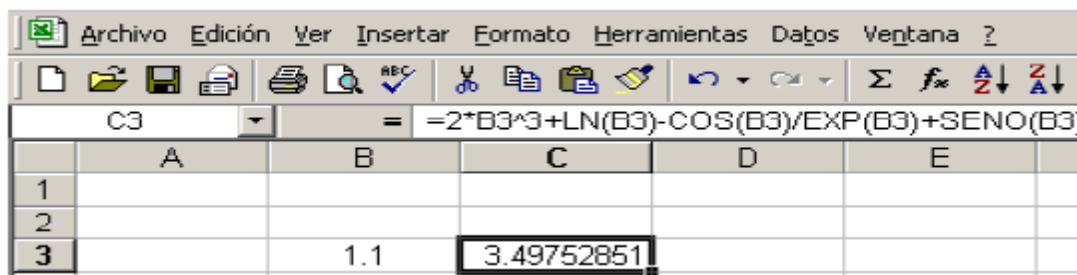


Figura 1.1 Evaluación de fórmulas.

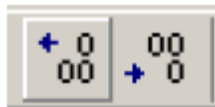
1.3.2 Errores comunes

Conforme nos vamos familiarizando con el uso de las formulas, van surgiendo algunos errores que usualmente son causados por un manejo inadecuado de la sintaxis o la incompatibilidad con la configuración de la computadora.

A continuación se describen los errores más frecuentes.

1. El valor de error #¿NOMBRE? aparece cuando Excel no reconoce texto en una formula. Para solucionar este conflicto se debe revisar la sintaxis e la formula o, si es una macro, verificar que esté en el módulo de la hoja en la que se está trabajando.
2. El valor de error #¡VALOR! Surge cuando se utiliza un tipo de argumento u operando incorrecto, por ejemplo, cuando evaluamos una función numérica en una celda que contiene algo que no sea un número (por defecto, el valor de una celda vacía es cero).
3. El valor de error #¡NUM! Aparece cuando existe un problema con algún número en una fórmula o función. Por ejemplo, si evaluamos una función logarítmica en cero o en un número negativo.
4. El valor de error #¡DIV/0! se produce cuando se divide una fórmula entre cero.
5. El valor de error #¡REF! se genera cuando una referencia a una celda no es válida.

6. Dependiendo de la forma en la que se encuentre configurado el sistema Windows, debe usarse punto o coma para separar la parte decimal de los números a evaluar. Para personalizarlo, se debe entrar al panel de control y en la Configuración regional se selecciona 'Números'. En la primera cajilla, 'Símbolo Decimal' se selecciona el punto o la coma, según sea el caso. Finalmente, se presiona el botón 'Aplicar' y luego 'Aceptar'.
7. Existe una circunstancia, que es comúnmente confundida con un error esta es cuando el sistema trabaja con poca precisión y se presentan valores numéricos no esperados. Por ejemplo, si el formato de una celda se ha definido para dos posiciones, entonces la operación $+1.999+1$ dará como resultado el valor de 2, que no es otra cosa que el resultado de la suma redondeado a dos decimales. El valor correcto se obtiene aumentando la precisión con el icono:



También se puede cambiar la precisión en el menú 'Formato-Celdas-Número-Posiciones decimales'.

Es importante recordar que estos cambios son solo en apariencia, pues, independientemente del número de dígitos que sean desplegados, Excel manipula números con una precisión de hasta 15 dígitos, si un número contiene más de 15 dígitos significativos, Excel los convertirá en ceros.

1.3.3 Evaluando una función con varios tipos de parámetros.

Muchas fórmulas a evaluar tienen argumentos de distinto tipo, pues algunos argumentos varían (a veces con un incremento determinado), mientras que otros permanecen constantes. Por lo general estos argumentos son tomados de celdas específicas, por lo que es importante saber manejar distintos escenarios para la evaluación de una función o formula.

1.3.3.1 Evaluación con argumentos variables

Continuando con el ejemplo que iniciamos en la sección 1.3.1, a partir de la celda B4 podemos continuar introduciendo valores, siempre en la columna B y con el cuidado de que los números ingresados no salgan del dominio de la función

$$f(x) = 2x^3 + \ln(x) - \frac{\cos(x)}{e^x} + \sin(x)$$

Que en este caso es el conjunto de los números reales positivos. Una vez hecho esto, se evalúa la función $f(x)$ en la celda C3, como se hizo previamente. Posteriormente, seleccionamos esta misma celda y ubicamos el puntero del mouse en la esquina inferior derecha, arrastrándolo hasta la celda deseada.

Otra opción es hacer un doble clic en la esquina inferior derecha de la celda a copiar y esto realiza la copia automáticamente, Figura 1.2.

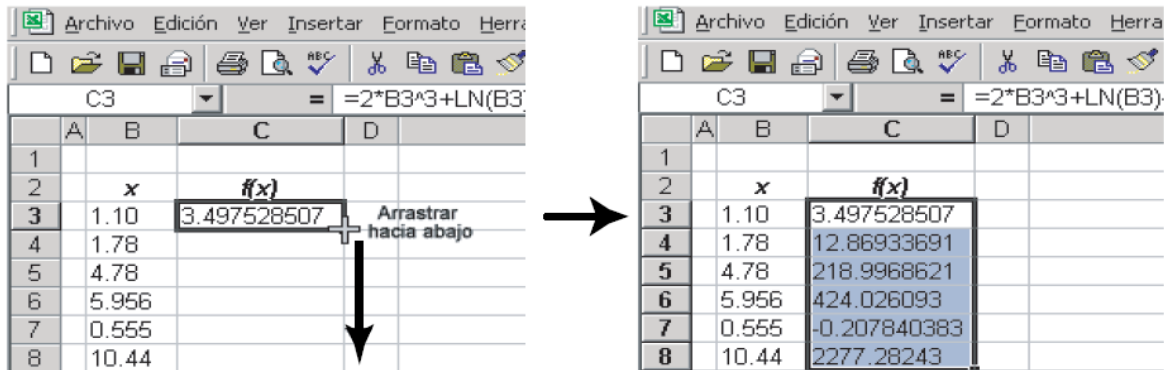


Figura 1.2. Copia de una fórmula en un grupo de celdas

1.3.3.2 Evaluación con argumentos variables y/o constantes

Es una práctica bastante común el tener que evaluar funciones o fórmulas que dependen de varios parámetros, algunos de los cuales se mantienen fijos mientras que otros son variables.

Ejemplo1.

El siguiente ejemplo describe una función con dos parámetros y una variable.

La función $P(t) = \frac{K}{1 + Ae^{-kt}}$ con $A = \frac{K-P_0}{P_0}$, describe el tamaño de una población en el momento t .

Dónde:

- k es una constante de proporcionalidad que se determina experimentalmente, dependiendo de la población particular que está siendo modelada,
- P_0 es la población inicial y
- K es una constante llamada *capacidad de contención* o *capacidad máxima que el medio es capaz de sostener*.

Si queremos evaluar $P(t)$ para diferentes valores de tiempo t en días, seguimos los siguientes pasos:

1. El primer paso consiste en escribir encabezados en cada una de las columnas (o filas) donde vamos a escribir los datos que serán argumentos de la función. Para este ejemplo vamos a comenzar en la celda B3 con las siguientes etiquetas.

P0 K k t P(t)

2. A continuación escribimos los valores de los parámetros, empezando en la celda B4

100 1000 0.08 0

3. Ahora escribimos la fórmula de la función $P(t)$ en la celda G4:

=C\$4/(1+((C\$4-B\$4)/B\$4)*EXP(-D\$4*E4))

Como se observa, el único argumento variable es t y nos interesa mantener a los demás argumentos constantes. Para mantener un valor constante, se le agrega el símbolo \$ antes del número de fila, como por ejemplo C\$4.

Para este ejemplo, como se muestra en la Figura 1.3, los argumentos constantes son los que están en las celdas B4, C4 y D4, mientras que el valor de t en la celda E4, es variable.

	A	B	C	D	E	F
1						
2						
3		P0	K	k	t	P(t)
4		100	1000	0,08	0	100
5					5	142,18925
6					6	152,229
7					10	198,2569
8					15	269,48745
9					17	302,11964
10					20	354,97892
11					21	373,50145
12					30	550,52086
13					45	802,62394

Figura 1.3 Evaluación con parámetros constantes y un parámetro con un incremento variable.

4. Por último, escribimos varios valores para t en la columna E, seleccionamos la celda F4 y arrastramos para evaluar $P(t)$ en el resto de valores de t .

Cabe mencionar que $P(t)$ es la solución de la llamada *ecuación logística*

$$\frac{dP}{dt} = kP \left(1 - \frac{P}{K} \right)$$

1.3.3.3 Construyendo rangos con un incremento fijo

Frecuentemente es necesario evaluar una función en una secuencia de valores igualmente espaciados, por lo que en la siguiente sección se explica cómo hacerlo, modificando el ejemplo previo de crecimiento de población.

1. Podemos seleccionar la columna C para poner los valores fijos P_0 , K , k , y el incremento h . En este caso, por ejemplo, $h=5$ servirá como incremento entre un tiempo y el siguiente, iniciando con $t=0$.
2. En la celda E4 escribimos el tiempo inicial $t=0$ y en la celda E5 se escribe el nuevo tiempo con el incremento h :

$$=+E4+C\$6$$

Debemos usar C\$6 para que el incremento se mantenga inalterado al copiar esta operación en otra celda situada en una fila diferente.

3. Ahora seleccionamos la celda E5 y la arrastramos hacia abajo para obtener los nueve tiempos con el respectivo incremento Figura 1.4.

	A	B	C	D	E	F
1						
2						
3		P_0	K	k	t	$P(t)$
4		100	1000	0,1	0	100
5					5	142,1893
6	Incremento:	$h=$	5		10	198,2569
7					15	269,4875
8					20	354,9789
9					25	450,8531
10					30	550,5209
11					35	646,291
12					40	731,6039
13					45	802,6239

Figura 1.4 Evaluación con un parámetro de incremento fijo.

Esto también se puede hacer, escribiendo en celdas consecutivas un valor y luego, el valor más el incremento, posteriormente se seleccionan ambas celdas y se arrastra el valor.

1.4 Creación de macros empleando VBA

1.4.1 ¿Qué es una macro?

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

1.4.2 Objetos, propiedades y métodos.

A la hora de trabajar con macros en Excel, deben tenerse claros ciertos conceptos de lo que se llama programación orientada a objetos (OOP). No nos extenderemos demasiado sobre la OOP, pero si definiremos a continuación los conceptos de Objeto, Propiedades y Métodos.

Objeto.

Cuando en el mundo real nos referimos a objeto significa que hablamos de algo abstracto que puede ser cualquier cosa. Por ejemplo podemos referirnos a objetos como coche, silla, casa, etc.

Cuando decimos que la clase coche representa a todos los coches del mundo significa que define como es un coche, cualquier coche. Dicho de otra forma y para aproximarnos a la definición informática, la clase coche define algo que tiene cuatro ruedas, un motor, un chasis,... entonces, cualquier objeto real de cuatro ruedas, un motor, un chasis,... es un objeto de la clase coche.

Propiedades.

Cualquier objeto tiene características o propiedades como por ejemplo el color, la forma, peso, medidas, etc. Estas propiedades se definen en la clase y luego se particularizan en cada objeto. Así, en la clase coche se podrían definir las propiedades Color, Ancho y Largo, luego al definir un objeto concreto como coche ya se particularizarían estas propiedades a, por ejemplo, Color = Rojo, Ancho = 2 metros y Largo = 3,5 metros.

Métodos.

La mayoría de objetos tienen comportamientos o realizan acciones, por ejemplo, una acción evidente de un objeto coche es el de moverse o lo que es lo mismo, trasladarse de un punto inicial a un punto final.

1.4.3 Conceptos útiles para trabajar con macros en Excel

- **Worksheet** (Objeto hoja de cálculo)
- **Range** (Objeto celda o rango de celdas).

Un objeto **Range** está definido por una clase donde se definen sus propiedades, recordemos que una propiedad es una característica, modificable o no, de un objeto. Entre las propiedades de un objeto **Range** están **Value**, que contiene el valor de la celda, **Column** y **Row** que contienen respectivamente la fila y la columna de la celda, **Font** que contiene la fuente de los caracteres que muestra la celda, etc.

Range, como objeto, también tiene métodos, recordemos que los métodos sirven llevar a cabo una acción sobre un objeto. Por ejemplo el método **Activate**, hace activa una celda determinada, **Clear**, borra el contenido de una celda o rango de celdas, **Copy**, copia el contenido de la celda o rango de celdas en el portapapeles.

- **Conjuntos.**

Una conjunto es una colección de objetos del mismo tipo . Por ejemplo, dentro de un libro de trabajo puede existir más de una hoja (Worksheet), todas las hojas de un libro de trabajo forman un conjunto, el conjunto Worksheets.

Cada elemento individual de un conjunto se referencia por un índice, de esta forma, la primera, segunda y tercera hoja de un libro de trabajo, se referenciarán por Worksheets(1), Worksheets(2) y Worksheets(3).

- **Objetos de Objetos.**

Es muy habitual que una propiedad de un objeto sea otro objeto. Siguiendo con el coche, una de las propiedades del coche es el motor, y el motor es un objeto con propiedades como caballos, número de válvulas, etc. y métodos, como aumentar_revoluciones, coger_combustible, mover_pistones, etc.

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). Tenga esto muy presente ya que utilizaremos frecuentemente Propiedades de un objeto que serán también Objetos.

Dicho de otra forma, hay propiedades que devuelven objetos, por ejemplo, la propiedad Range de un objeto Worksheet devuelve un objeto de tipo Range.

- **Programación Orientada a Objetos o Programación Basada en Objetos.**

Hay una sutil diferencia entre las definiciones del título. Programación orientada a Objetos, significa que el programador trabaja con objetos fabricados por él mismo, es decir, el programador es quien implementa las clases para luego crear objetos a partir de ellas. Lo que haremos nosotros, por el momento, será utilizar objetos ya definidos por la aplicación Excel (WorkSheets, Range,...) sin implementar ninguno de nuevo, por lo que en nuestro caso es más correcto hablar de programación basada en objetos. Observe que esta es una de las grandes ventajas de la OOP, utilizar objetos definidos por alguien sin tener que conocer nada sobre su implementación, sólo debemos conocer sus propiedades y métodos y utilizarlos de forma correcta.

1.4.4 Grabar una macro

En Excel existen dos maneras de crear una macro, la primera es usando la herramienta para grabar macros que trae Excel y la segunda es empleando el lenguaje VBA.

En esta sección explicaremos como usar el primer método.

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.

1.4.4.1 Mi primera Macro.

En este apartado crearás una macro y te explicaremos como acceder a ella y como parametrizarla.

Activación de la barra de herramientas

Antes de empezar es necesario colocar la barra de herramienta de Programador, que contendrá todas las herramientas necesarias para realzar todos los ejercicios y prácticas de este curso.

Vamos a Inicio>Opciones Figura 1.6.

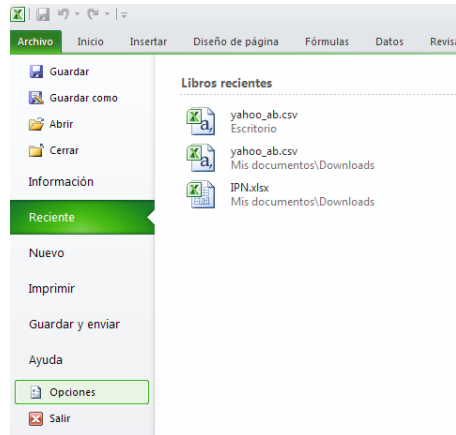


Figura 1.6 Menú opciones.

Posteriormente en la ventana emergente que se despliega se busca Personalizar cinta de opciones y se selecciona en el menú del lado derecho la opción de Programador Figura 1.7.

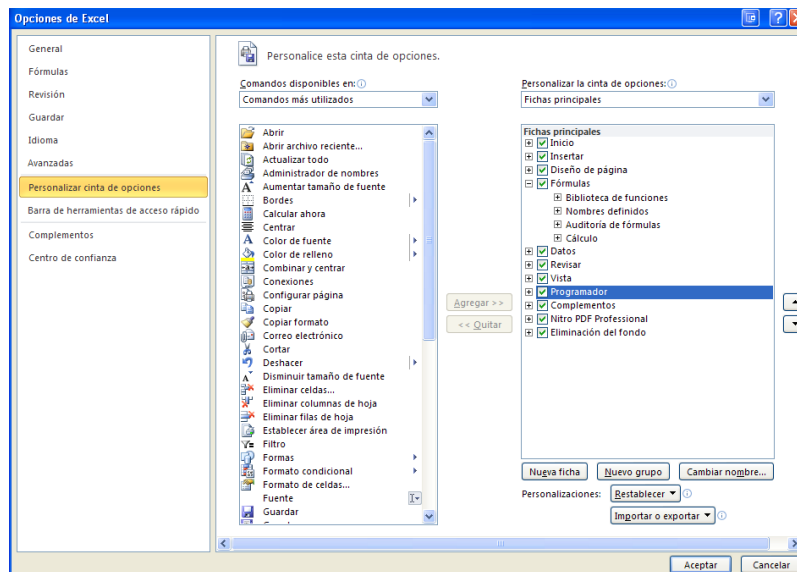


Figura 1.7. Menú opciones2.

Posteriormente en la barra de herramientas aparecerá la pestaña de Programador
Figura 1.8.

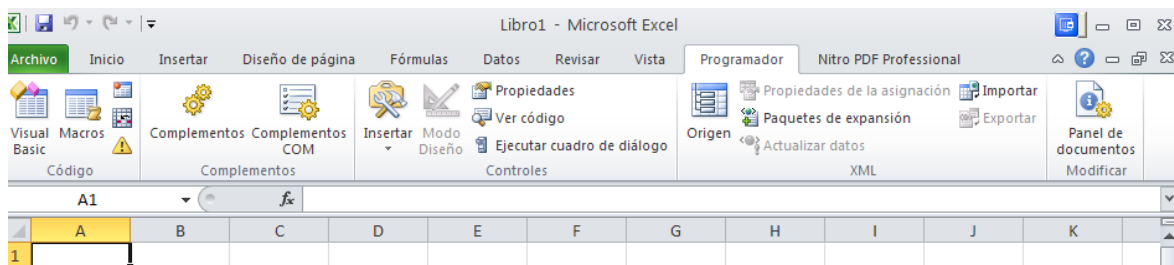



Figura 1.8. Menú Programador.

Creando una macro

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 Programador y el icono .
4. En el cuadro de diálogo en la caja de texto de Nombre de la macro² escriba: “miPrimerMacro”, seguido en el cuadro de texto para Método Abreviado escriba “m”.

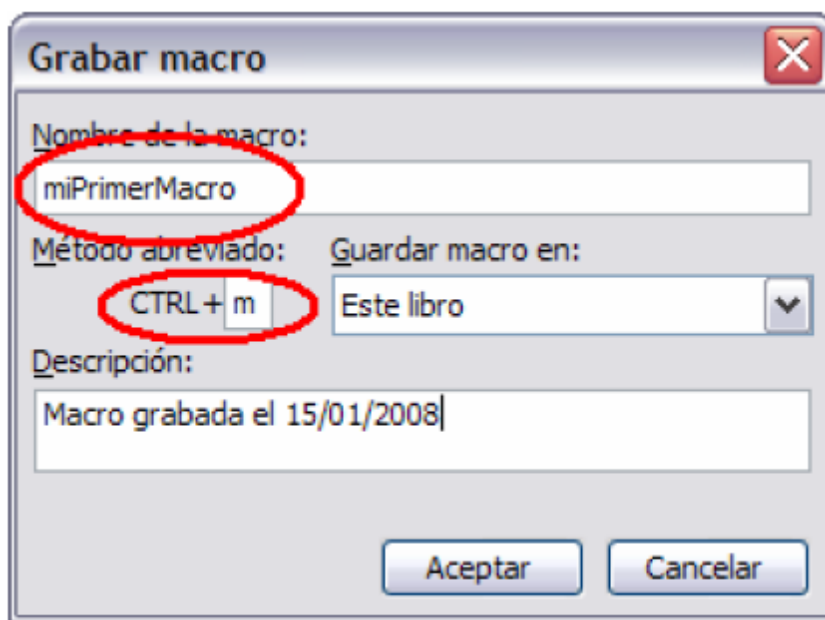


Figura 1.9 Grabar macro.

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.

6. Presione el botón detener (el cuadro azul de la Figura 2.0). Se ha acabado de grabar su primer macro.

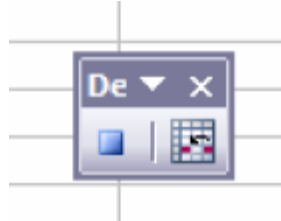


Figura 2.0 Detener.

1.4.4.2 Ejecutar una macro

Para ejecutar una macro existen varias formas, la primera es ir a la barra de menú <Programador/ Macros...> dar clic en Macros... y aparecerá un cuadro de diálogo como la Figura 2.1 en seguida seleccione la macro que desea ejecutar y haga clic en el botón <Ejecutar>.

La otra forma es, usar el método abreviado asignado sólo es necesario presionar la tecla “ctrl” y sin dejar de presionarla la letra “m”, esta acción se denotará en lo siguiente con “ctrl+m”.

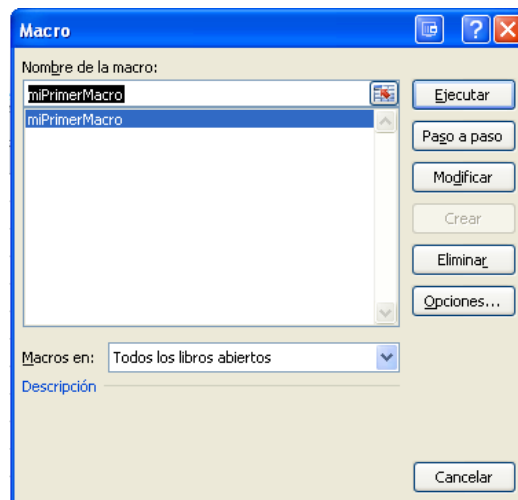


Figura 2.1 Macro.

Practica 1 “Grabar Macros”

Objetivo: Que el participante tenga nociones claras del uso de las macros, empleando algunas de las herramientas que brinda el menú Programador de Excel.

Instrucciones: Crear las siguientes macros, en el orden que se indica.

- 1) Macro “archivo” que se active con el método abreviado de su elección y que permita abrir un archivo.
- 2) Macro “selección” que seleccione las celdas A1:A10 de la primera hoja del libro abierto.
- 3) Macro “letra” que ponga tipo de letra arial
- 4) Macro “color” que ponga letra de color azul.
- 5) Macro “tamaño” que ponga letra de tamaño 14.
- 6) Macro “Todo” que una las macros anteriores.

1.4.5 Manipulación de macros

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. Por ejemplo, si la macro de ajuste de texto también tiene que aplicar el formato de negrita al texto, se puede grabar otra macro para aplicar el formato de negrita a una celda y, a continuación, copiar las instrucciones de esa macro a la macro de ajuste de texto. 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.

Para acceder al código fuente de la macro, solamente es necesario seleccionar en el menú Programador > Macro > Modificar, seleccionando la macro de la cual queremos ver el código Figura 2.2 y mostrará una ventana como en la Figura 2.3 con el código generado al momento de grabar la macro.

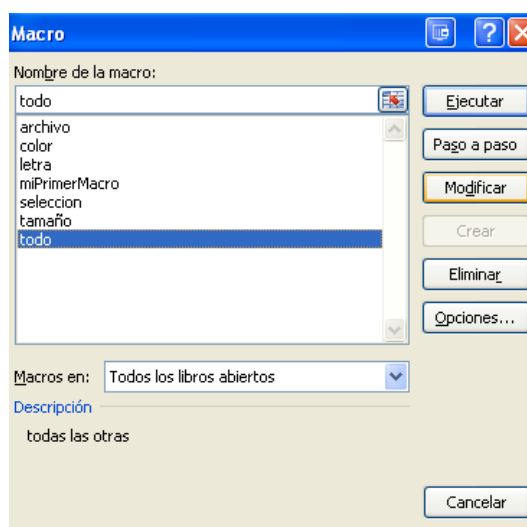


Figura 2.2 Manipulación de macros.

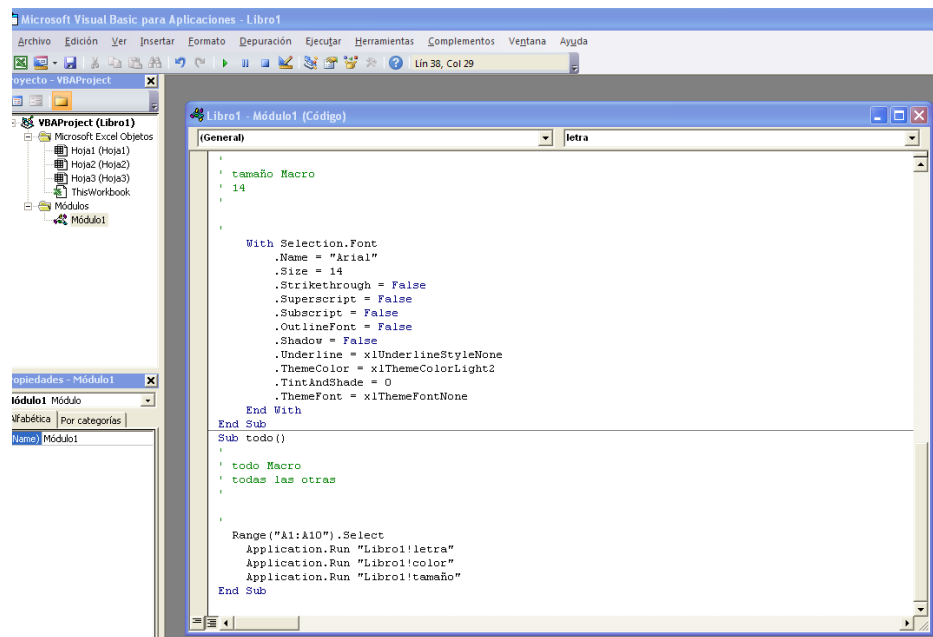


Figura 2.3 Editor de VBA.

1.4.6 Editor de Visual Basic

El editor de Visual Basic es la aplicación que utilizaremos para construir las macros que interactuarán junto con los libros de trabajo. A continuación prepararemos un archivo en el que escribiremos las primeras instrucciones en Visual Basic.

Preparar un archivo nuevo.

Para entrar en el editor de Visual Basic, ejecute los siguientes pasos.

1.-En el menú Programador seleccione la opción Visual Basic Figura 2.4. Y se abrirá la ventana de la Figura 2.5.

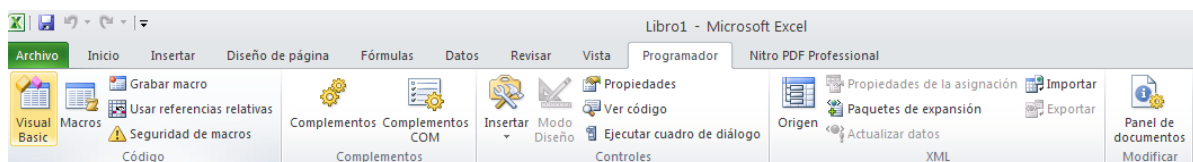


Figura 2.4 Acceso al editor VBA.

Maximice la ventana para trabajar más cómodamente y procure tener activadas las ventanas Explorador de proyectos y la ventana Propiedades (Ver/Explorador de proyectos y Ver/Ventana propiedades).

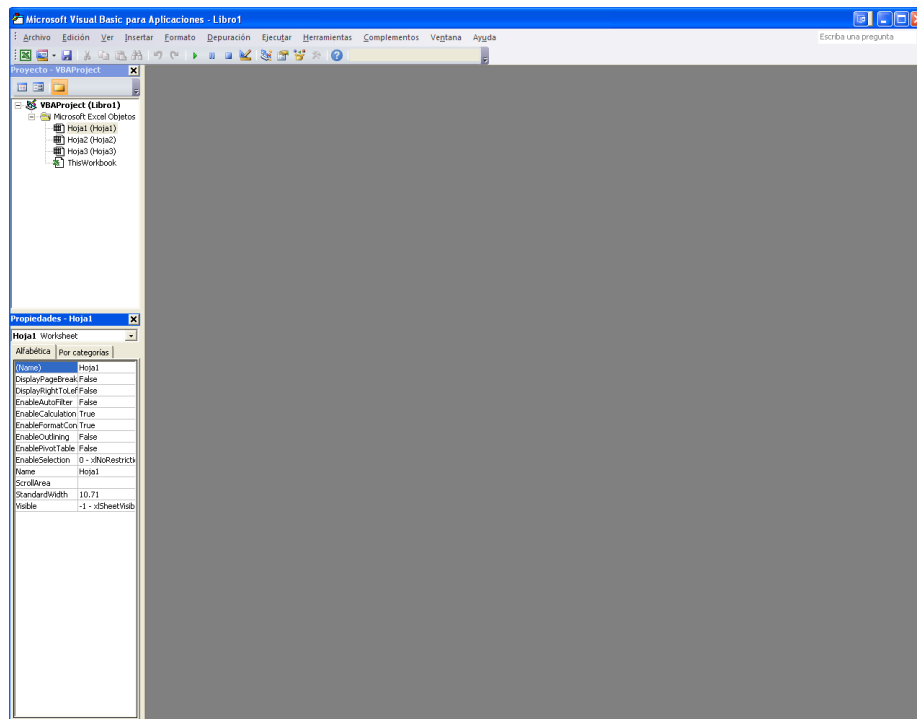


Figura 2.5 Editor VBA.

Insertar un nuevo módulo

Un módulo sirve para agrupar procedimientos y funciones. El procedimiento y la función son entidades de programación que sirven para agrupar instrucciones de código que realizan una acción concreta.

Para insertar un módulo active opción del menú **Insertar/ Módulo**. Se activará una nueva ventana, si aparece demasiado pequeña, maximícela.

Insertar procedimientos

Ya hemos dicho que un procedimiento es un bloque de instrucciones de código que sirven para llevar a cabo alguna tarea específica. Un procedimiento empieza siempre con la instrucción

Sub *Nombre_Procedimiento*

Y termina con la instrucción

End Sub.

A continuación crearemos un procedimiento para poner el texto "Hola" en la casilla A1.

Ejemplo 1

Sub *Primero*

Range("A1").**Value** = "Hola"

End Sub

Observe el código.

Range("A1").**Value**="Hola"

En esta línea se indica que trabajamos con un objeto **Range**. Para indicarle que nos referimos a la casilla A1, encerramos entre paréntesis esta referencia.

De este objeto, indicamos que se quiere establecer un nuevo valor para la propiedad **Value**, observe que para separar el objeto de su propiedad utilizamos la notación punto.

Recuerde que el conjunto **Range** es un objeto que pende del objeto **Worksheets**, así por ejemplo el siguiente código haría lo mismo que el anterior.

Worksheets(1).**Range**("A1").**Value** = "Hola"

Bueno, de hecho no hace lo mismo, en la primera opción, el texto "Hola" se pone dentro de la casilla A1 de la hoja activa, mientras que en el segundo es en la casilla A1 de primera hoja (del conjunto de hojas).

La segunda notación es más larga, pero también más recomendable ya que se especifican todos los objetos. En muchas ocasiones se pueden omitir algunos objetos precedentes, no es aconsejable hacerlo, debido a que sus programas perderán claridad y concisión.

Si desea hacer referencia a la hoja activa puede utilizar **ActiveSheet**, así, el primer ejemplo lo dejaremos de la manera siguiente.

Sub *Primero*

ActiveSheet.**Range**("A1").**Value** = "Hola"

End Sub

Si desea poner "Hola" (o cualquier valor) en la casilla activa, puede utilizar la propiedad (objeto) **Activecell** de **Worksheets**. Así para poner "Hola" en la casilla activa de la hoja activa sería

Sub *Primero*

ActiveSheet.ActiveCell.Value = "Hola"

End Sub

Para terminar con este primer ejemplo. **Worksheets** están dentro del Objeto **WorkBooks** (libros de trabajo) y **WorkBooks** están dentro de **Application**. **Application** es el objeto superior, es el que representa la aplicación Excel. Así, el primer ejemplo, siguiendo toda la jerarquía de objetos quedaría de la forma siguiente.

Sub *Primero*

Application.WorkBooks(1).Worksheets(1).Range("A1").
Value = "Hola"

End Sub

Insistiendo con la nomenclatura, **Application** casi nunca es necesario especificarlo, piense que todos los objetos penden de este, **WorkBooks** será necesario implementarlo si en las macros se trabaja con diferentes libros de trabajo (diferentes archivos), a partir de **Worksheets**, es aconsejable incluirlo en el código, sobre todo si se quiere trabajar con diferentes hojas, verá, sin embargo, que en muchas ocasiones no se aplica.

Ejecutar un procedimiento o función.

Pruebe ejecutar el primer procedimiento de ejemplo.

1. Sitúe el cursor dentro del procedimiento.
2. Active opción de la barra de menús **Ejecutar/ Ejecutar Sub Userform**. También puede hacer clic sobre el botón o pulsar la tecla **F5**.

Practica 2 “Observando los códigos de una macro en Excel”

Objetivo: El asistente interactuar con códigos generados por el editor VBA al momento de grabar una macro.

Instrucciones: Gravar las siguientes macros, en el orden que se indica.

- 1) Generar una macro que escriba un nombre en una celda y lo ponga cursivas y observe el código.
- 2) Generar una macro que escriba un nombre en una celda y lo centre y observe el código.
- 3) Generar una macro que escriba un nombre en una celda y cambie el tamaño de letra a 18 y el tipo a Arial y observe el código.
- 4) Generar una macro que ejecute a todas las anteriores y observe el código.

Códigos más comunes

Trasladarse a una Celda

```
Range("A1").Select
```

Escribir en una Celda

```
Activecell.FormulaR1C1="Paty"
```

Letra Negrita

```
Selection.Font.Bold = True
```

Letra Cursiva

```
Selection.Font.Italic = True
```

Letra Subrayada

```
Selection.Font.Underline = xlUnderlineStyleSingle
```

Centrar Texto

```
With Selection
```

```
.HorizontalAlignment = xlCenter
```

```
End With
```

Alinear a la izquierda

```
With Selection
```

```
.HorizontalAlignment = xlLeft
```

```
End With
```

Alinear a la Derecha

```
With Selection
```

```
.HorizontalAlignment = xlRight
```

```
End With
```

Tipo de Letra (Fuente)

```
With Selection.Font
```

```
.Name = "AGaramond"
```

```
End With
```

Tamaño de Letra (Tamaño de Fuente)

```
With Selection.Font
```

```
.Size = 15
```

```
End With
```

Copiar

```
Selection.Copy
```

Pegar

```
ActiveSheet.Paste
```

Cortar

Selection.Cut

Ordenar Ascendente

Selection.Sort Key1:=Range("A1"), Order1:=xlAscending, Header:=xlGuess, _
OrderCustom:=1, MatchCase:=False, Orientation:=xlTopToBottom

Orden Descendente

Selection.Sort Key1:=Range("A1"), Order1:=xlDescending, Header:=xlGuess, _
OrderCustom:=1, MatchCase:=False, Orientation:=xlTopToBottom

Buscar

Cells.Find(What:="Paty", After:=ActiveCell, LookIn:=xlFormulas, LookAt _
:=xlPart, SearchOrder:=xlByRows, SearchDirection:=xlNext, MatchCase:= _
False).Activate

Insertar Fila

Selection.EntireRow.Insert

Eliminar Fila

Selection.EntireRow.Delete

Insertar Columna

Selection.EntireColumn.Insert

Eliminar Columna

Selection.EntireColumn.Delete

Abrir un Libro

Workbooks.Open Filename:="C:\Mis documentos\video safe 3.xls"

Grabar un Libro

ActiveWorkbook.SaveAs Filename:="C:\Mis documentos\piscis.xls", FileFormat _
:=xlNormal, Password:="", WriteResPassword:="", ReadOnlyRecommended:= _
False, CreateBackup:=False

1.4.7 Cuadro de control

Una de las opciones más interesantes que tiene el Excel es la de utilizar los "cuadros de control".

Los cuadros de control se usan para crear verdaderos programas en Excel y pueden ser de mucha utilidad.

Para poder visualizar esta barra es necesario dirigirse a Programador > Insertar mostrando el contenido de la Figura 2.6.

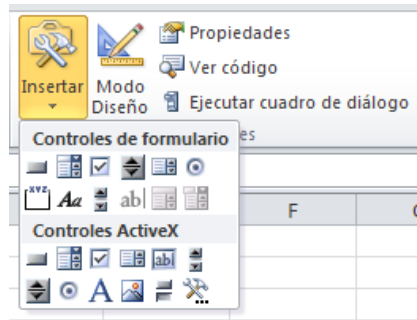


Figura 2.6 Cuadro de control.

Como podrá ver está dividida en Tres secciones
La primera consta de:

- Modo diseño
- Propiedades
- Ver código

La segunda tiene todos los botones y herramientas que se pueden usar.
La tercera se utiliza para agregar o quitar botones con lo que expande mas las posibilidades de esta barra.

Control numérico.

Elija “Control de Número” 

Dibuje en la planilla el cuadro donde estará situado el control. Por Ej.:



Seleccione propiedades 

Aquí podemos cambiar entre otras cosas.


Max: El número máximo que recorrerá este cuadro


Min: El número mínimo

LinkedCell: Celda donde se va a mostrar el número relacionado con el cuadro de control: Por ejemplo en la celda A4

Cierre las “propiedades” último paso y el más importante es salir del modo diseño dando clic en “Modo Diseño”.

Si es presionada la flecha hacia la derecha el número en la celda A4 irá aumentando y si es presionada la de la izquierda irá decreciendo.

Una de las partes más fundamentales es “Modo Diseño”  es la posibilidad de cambiar cualquier cosa del Control, por ejemplo tamaño, ubicación, etc. Y para poder probar su funcionamiento se debe salir de “Modo Diseño”.

Otra parte fundamental es “Propiedades” , donde es posible manipular atributos fundamentales para que el control se adapte a sus necesidades. Por ejemplo, Max, Min, LinkedCell, Delay (velocidad con que cambian los números), Shadow (le agrega sombra al control), entre otras más.

Ejemplo 2

Otro de los botones más útiles que tiene la Barra de herramientas de “Cuadro de Controles” es el “Cuadro Combinado”.

Primera Sección:

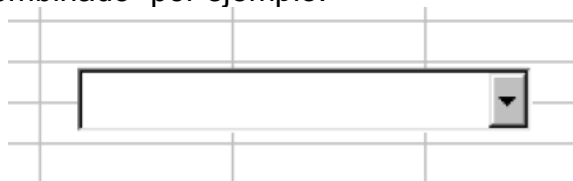
Escribir en la hoja3 los siguientes datos de manera consecutiva.


- Televisor
- Heladera
- Licuadora
- Monitor
- Teclado

Nombre a este rango de datos como “datos” (sin comillas) (Para nombrar un rango de datos marque los datos y escriba el nombre en el “Cuadro de nombres”)

Segunda parte:

Dibuje un “Cuadro combinado” por ejemplo:



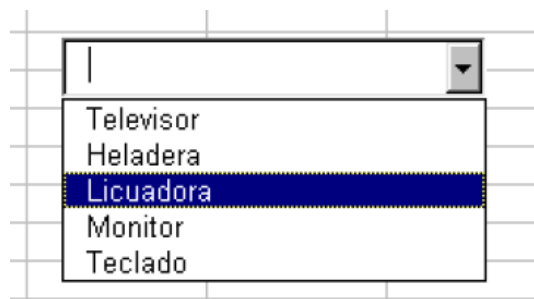
Seleccione propiedades 

Busque la propiedad: ListFillRange y escriba: “datos” (sin comillas).

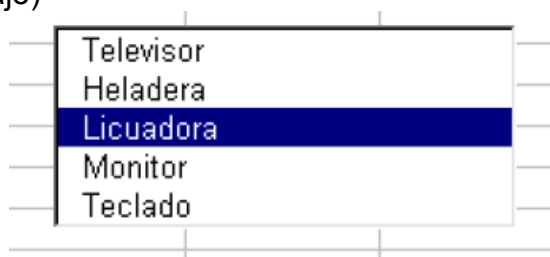
En la propiedad LinkedCell Escriba A1 (es la celda donde se mostrará el dato elegido).

Salga del “Modo Diseño” y pruebe este ejemplo:

Podrá comprobar que en el “Cuadro combinado” se encuentran los datos ingresados en la hoja3 y que cuando elige uno, éste se muestra en la celda A1.



Es necesario recordar que esto se aplica tanto a “Cuadro combinado” (arriba) o a “Cuadro de lista” (abajo)

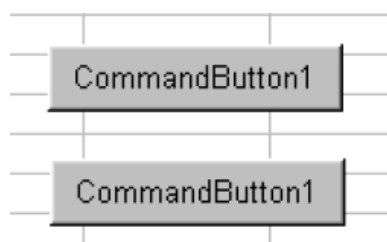


Ejemplo 3

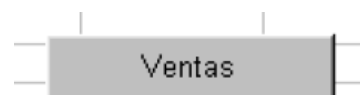
Esta vez haremos un botón que cuando se presione pase a otra hoja del Excel. Por ejemplo se puede hacer un menú con varios botones que al presionarlos pasen a las distintas opciones.

En la Hoja1 crear dos “botones de comando”.

Por Ejemplo:



Seleccione el primero botón y muestre las propiedades
Cambie la Propiedad “Caption” por : “Ventas”




Seleccione el segundo botón y muestre las propiedades
Cambie la Propiedad “Caption” por: “Compras”



Seleccione el primer botón y haga clic en ver código 

En esta parte se abrirá el Editor de Visual Basic y debe escribir lo siguiente:

Hoja2.activate

Cierre el editor de Visual Basic y seleccione el segundo botón, haga clic en ver código  y escriba:

Hoja3.activate

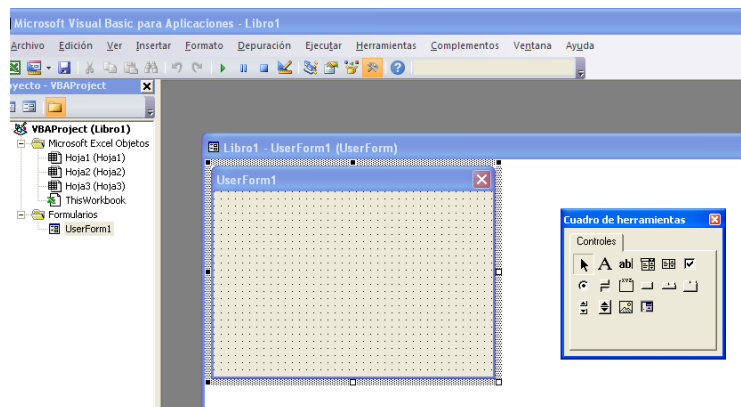
Creando formularios y programándolos

Un formulario es una ventana que se programa por medio de controles, estos controles responden a eventos que son programados dentro del ambiente de VBA.

En esta sección se muestra como crear un formulario y programar cada objeto:

Ejemplo 4

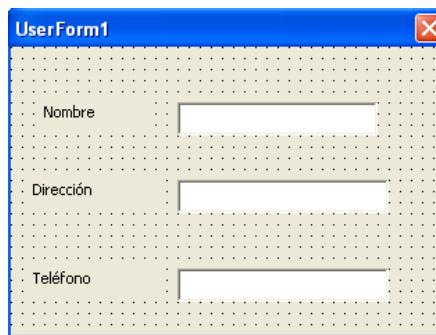
1. Abrir el editor de Visual Basic.
2. Activar las opciones:
Explorador de Proyectos
Ventana Propiedades
3. Del Menú Insertar elegir la opción UserForm. En el Explorador de Proyecto se observara que se insertó el UserForm.



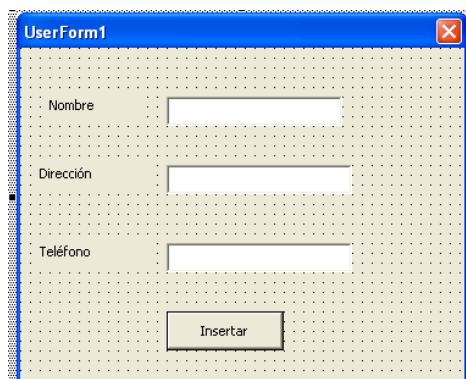
4. Elegir del Cuadro de Herramientas el Control Etiqueta y arrastrar al UserForm, dibujando en el Formulario UserForm1 la etiqueta Label1, después de un clic en la etiqueta dibujada y podrá modificar el nombre de adentro y pondremos Nombre. Si por error da doble clic en la etiqueta y lo manda a la pantalla de programación de la etiqueta, solo de doble clic en UserForm1 que se encuentra en el Explorador de Proyecto.

5. Elegir del Cuadro de Herramientas el control Cuadro de Texto y arrastrar dibujando en el formulario UserForm1 el cuadro de texto a un lado de la etiqueta Nombre. El cuadro de texto debe de estar vacío y su nombre será Textbox1, el nombre solo aparecerá en el control.

6. Haga los dos pasos anteriores igualmente poniendo Dirección en la Label2 y Teléfono en la Label3 y también dibuje su respectivo Textbox.

A screenshot of a VBA UserForm titled 'UserForm1'. It contains three labels: 'Nombre', 'Dirección', and 'Teléfono', each followed by an empty text box. The form has a blue title bar and a standard Windows-style border.

8. Posteriormente elegir del Cuadro de Herramientas el control Botón de Comando y Arrastrar dibujando en el Formulario UserForm1 el Botón, después de un clic en el nombre del Botón para modificar el nombre que será Insertar.

A screenshot of the same 'UserForm1' as before, but now with a command button labeled 'Insertar' positioned below the 'Teléfono' text box. The button has a standard Windows button appearance with a border and a shadow.

9. Ahora de doble clic sobre el control Textbox1 para programarlo y después se inserta el siguiente código:

```
Private Sub TextBox1_Change()  
Range("A9").Select  
ActiveCell.FormulaR1C1 = TextBox1  
End Sub
```

Esto indica que se vaya a **A9** y escriba lo que hay en el **Textbox1**.

10. Ahora de doble clic sobre el control Textbox2 para programarlo y después se inserta el siguiente código:

```
Private Sub TextBox2_Change()  
Range("B9").Select  
ActiveCell.FormulaR1C1 = TextBox2  
End Sub
```

Esto indica que se valla a **B9** y escriba lo que hay en el **Textbox2**.

11. Ahora de doble clic sobre el control Textbox3 para programarlo y después se inserta el siguiente código:

```
Private Sub TextBox3_Change()  
Range("C9").Select  
ActiveCell.FormulaR1C1 = TextBox3  
End Sub
```

Esto indica que se valla a **C9** y escriba lo que hay en el **Textbox3**.

12. Ahora de doble clic sobre el Botón de Comando para programarlo y después se inserta el siguiente código:

```
Private Sub CommandButton1_Click()  
    Rem inserta un renglón  
    Selection.EntireRow.Insert  
    Rem Empty Limpia Los Textbox  
    TextBox1 = Empty  
    TextBox2 = Empty  
    TextBox3 = Empty  
    Rem TextBox1.SetFocus Envía el cursor al Textbox1 para  
    volver a capturar los datos  
    TextBox1.SetFocus  
End Sub
```

El comando Rem es empleado para poner comentarios dentro de la programación, el comando Empty es empleado para vaciar los Textbox.

13. Ahora presione el botón Ejecutar User/Form que se encuentra en la barra de herramientas o simplemente la tecla de función F5.
Se activara el Userform1 y todo lo que escriba en los Textbox se escribirá en Excel y cuando se presione el botón Insertar, se insertará un renglón y se vaciarán los Textbox y después se mostrará el cursor en el Textbox1.

Empleando formulas

Es de suma importancia saber aplicar formulas con macros de Excel, ya que la mayoría de las hojas de cálculos las involucran, por ejemplo los Inventarios, las Nóminas o cualquier otro tipo de hoja las llevan, es por eso que en la siguiente sección se muestra cómo conjuntar formulas con macros de Excel.

Ejemplo 5.

1. Retomando los pasos del ejemplo anterior, generaremos un formulario con los siguientes campos:
 - a. Nombre
 - b. Edad
 - c. Días vividosY un botón de Resultado.

Los datos que se preguntaran serán Nombre y Edad, los Días Vividos se generaran automáticamente cuando insertes la edad. A continuación se muestra como se deben de programar estos Controles:

Programación de los Controles:

Botón.

```
Private Sub CommandButton1_Click()  
    Selection.EntireRow.Insert  
    TextBox1 = Empty  
    TextBox2 = Empty  
    TextBox3 = Empty  
    TextBox1.SetFocus  
End Sub
```

Nombre.

```
Private Sub TextBox1_Change()  
    Range("A9").Select  
    ActiveCell.FormulaR1C1 = TextBox1  
End Sub
```

Edad.

```
Private Sub TextBox2_Change()  
    Range("B9").Select  
    ActiveCell.FormulaR1C1 = TextBox2  
    Rem aquí se crea la Formula  
    TextBox3 = Val(TextBox2) * 365  
    Rem El Textbox3 guardara el total de la multiplicación del Textbox2 por 365  
    Rem El Comando Val permite convertir un valor de Texto a un Valor Numérico  
    Rem Esto se debe a que los Textbox no son Numéricos y debemos de Convertirlos  
End Sub
```

Días vividos.

```
Private Sub TextBox3_Change()  
    Range("C9").Select  
    ActiveCell.FormulaR1C1 = TextBox3  
End Sub
```

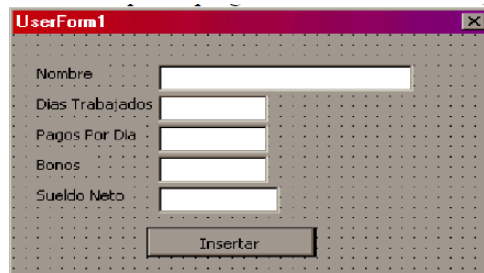
Al ejecutarse el formulario y se introduzca la edad se obtendrán el resultado de los días vividos en el Textbox3 y también en Excel. El comando Val es un comando de Visual Basic que te permite convertir un valor de texto a un valor numérico.

Ejemplo 6.

Generaremos otro ejemplo, Crea el Siguiete Formulario con los siguientes datos:

- 5 Etiquetas
- 5 Textbox
- 1 Botón de Comando

Los datos que se preguntaran serán Nombre, Días Trabajados, Pago por Día, Bonos y Sueldo Neto.



Genera el siguiente código:

```
Private Sub CommandButton1_Click()  
    Selection.EntireRow.Insert  
    TextBox1 = Empty  
    TextBox2 = Empty  
    TextBox3 = Empty  
    TextBox1.SetFocus  
End Sub  
  
Private Sub TextBox1_Change()  
    Range("A9").Select  
    ActiveCell.FormulaR1C1 = TextBox1  
End Sub
```

```
Private Sub TextBox2_Change()
    Range("B9").Select
    ActiveCell.FormulaR1C1 = TextBox2
End Sub
```

```
Private Sub TextBox3_Change()
    Range("C9").Select
    ActiveCell.FormulaR1C1 = TextBox3
End Sub
```

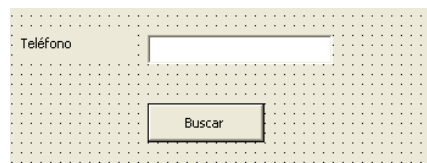
```
Private Sub TextBox4_Change()
    Range("D9").Select
    ActiveCell.FormulaR1C1 = TextBox4
    Rem aquí se crea la formula
    TextBox5 = Val(TextBox2) * Val(TextBox3) + Val(TextBox4)
    Rem El TextBox5 guardara el total
End Sub
```

```
Private Sub TextBox5_Change()
    Range("E9").Select
    ActiveCell.FormulaR1C1 = TextBox5
End Sub
```

Cuando se introduzca el Bonos automáticamente se generara el Sueldo Neto.

Buscando información

Se puede buscar información con un Textbox programándolo de la siguiente forma:



```
Private Sub TextBox1_Change()
    Range("a9").Select
    ActiveCell.FormulaR1C1 = TextBox1
End Sub
```

```
Private Sub CommandButton1_Click()
    Cells.Find(What:=TextBox1, After:=ActiveCell, LookIn:=xlFormulas, LookAt _
:=xlPart, SearchOrder:=xlByRows, SearchDirection:=xlNext, MatchCase:= _
False).Activate
End Sub
```


Practica 3 “Formulario”

Objetivo: Que el participante realice una factura empleando el lenguaje de programación VBA en Excel.

Instrucciones: Crear un formulario para generación de una factura digital.

- 1) Dibujar un UserForm
- 2) Colocar una etiqueta para cada uno de los siguiente rubros, con su respectivo Textbox y un Botón de Comando en el UseForm.
 - a. Número de factura (generación automática)
 - b. Nombre del cliente
 - c. Producto
 - d. Precio del producto
 - e. Número de productos
 - f. Subtotal (generación automática)
 - g. Descuento
 - h. I.V.A (generación automática)
 - i. Precio Total (generación automática)
- 3) Usar los comandos necesarios para programar cada objeto.
- 4) Mostrar el resultado final en el formulario y en Excel.

Bibliografía

1. Ayuda de Microsoft Excel 2010.
2. <http://office.microsoft.com/es-hn/infopath/CH011097053082.aspx>
Sitio oficial de Microsoft que presenta varios enlaces a artículos relacionados con la Validación de datos en Excel.
3. <http://office.microsoft.com/es-hn/excel/HA010346573082.aspx>
Sitio con ejemplos sencillos acerca de la validación de datos en Excel.
4. <http://office.microsoft.com/es-es/excel/HP100725993082.aspx>
Sitio en línea de Microsoft para el área de Excel que presenta ejemplos e información detallada que incluye las diferentes versiones de Excel.
5. <http://office.microsoft.com/es-hn/infopath/CH011097053082.aspx>
Sitio oficial de Microsoft que presenta varios enlaces a artículos relacionados con funciones en Excel.
6. <http://www.uv.mx/iip/enrique/sistemasII/apuntesexcel.pdf>
Sitio del Instituto Tecnológico Autónomo de México en donde se encuentran las generalidades de Excel y reglas para el uso de las bibliotecas de funciones.
8. <http://www.eumed.net/libros/finanzas.htm>
Sitio encontraras libros gratuitos con funciones financieras.
9. <http://office.microsoft.com/>
Sitio en línea de donde es posible consultar información más detallada sobre cualquier tópico de MS Excel.
10.
<http://office.microsoft.com/eses/excel/HP052047113082.aspx?pid=CH062528393082>
Sitio que contiene información relaciona con las Macros en Excel
11. <http://support.microsoft.com/kb/213740/es>
Sitio que contiene información relaciona con las Macros en Excel
12.
http://ciberconta.unizar.es/leccion/cursointermedioexcel/01_macro/macro03.htm
Sitio que contiene información relaciona con las Macros en Excel