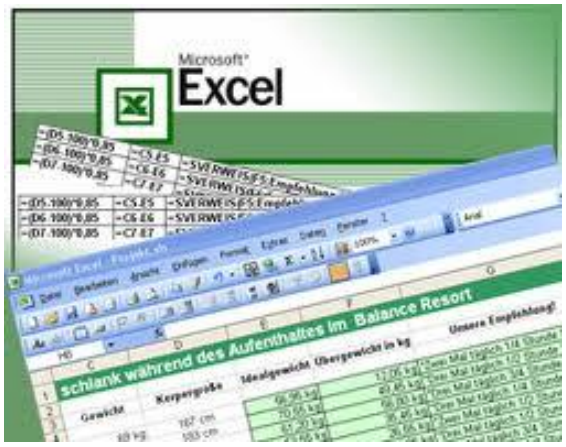
PROGRAMACION DE MACROS EN EXCEL



RicoSoft
INFORMÁTICA PROFESIONAL

Aviso legal: todos los textos y pantallas están extraidos del curso de macros de aulaclic.

Curso de Programación de Macros en Excel
Índice del curso

Índice detallado

Unidad 1. Las herramientas del Excel

Introducción

Antes de ingresar al desarrollo del Lenguaje Visual Basic para Aplicaciones, haremos una breve exposición de algunas herramientas del Excel que se supone son conocidas por nuestro amable lector. Sin embargo, para unificar criterios y con el ánimo de recordar lo conocido, haremos una exposición de la secuencia de pasos que se sigue para ejecutar o usar determinadas herramientas.

Este repaso nos servirá también para mostrar dicha secuencia usado en las versiones Excel 2003 y Excel 2007.

El objetivo de esta sección es, entonces, refrescar al participante de dichos procedimientos a fin de que pueda usarlos sea durante la grabación de macros o cuando tenga que programar ciertas acciones que requieran del uso de tales procedimientos.

En el numeral 1 presentamos la secuencia de pasos para realizar o ejecutar la herramienta con algunas observaciones y comentarios. En el numeral 2 daremos algunos ejemplos de grabación de macros que hagan uso de determinadas herramientas. En el numeral 3 desarrollaremos algunas macros que nos permitan realizar algunas operaciones en Excel de uso frecuente. Esta introducción termina con una breve exposición de la filosofía de objetos en la Programación Orientada a Objetos (POO), técnica empleada en la programación de macros.

Filtro avanzado

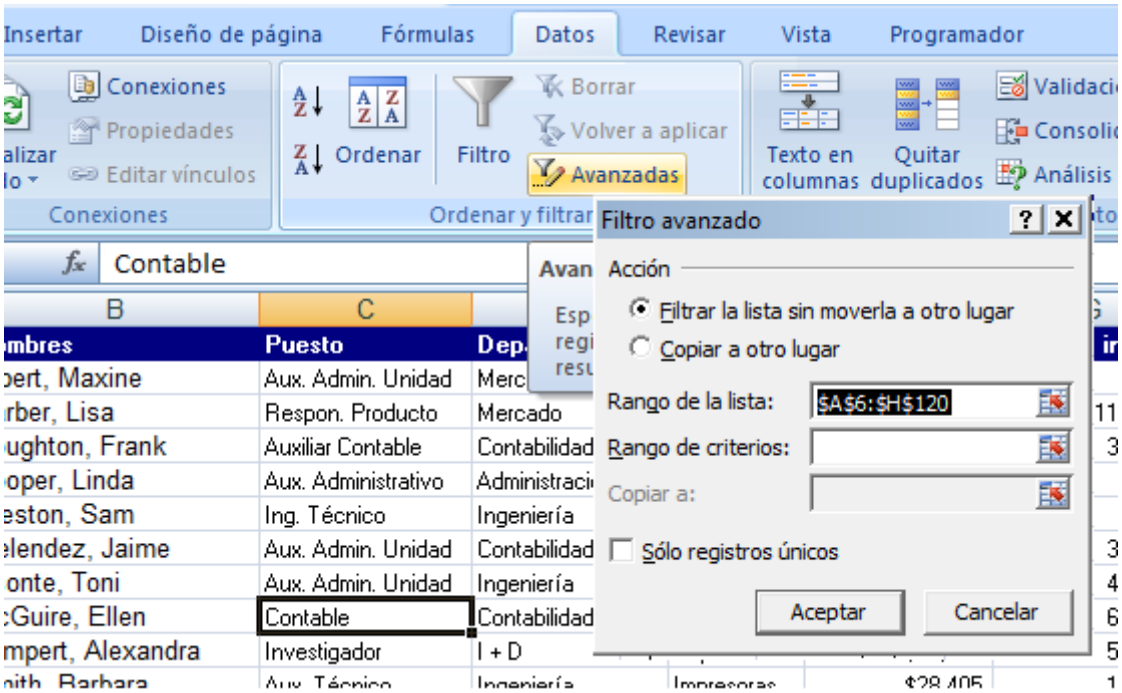


Figura 1

Usamos filtro avanzado para poder extraer, sea en una misma hoja o en otra, ciertos registros de una base de datos que cumplen determinadas condiciones.

Para realizar el filtro avanzado es necesario definir previamente el rango de criterios así como conocer el rango de datos y la celda, a partir de la cual se deben emitir los resultados.

El rango de criterios debe contener en su primera fila, los nombres de los campos (columnas) y en las siguientes filas contener los valores o criterios del filtrado.

La imagen que se presenta en la Figura 01, nos muestran la secuencia de pasos que se debe seguir para realizar el filtro avanzado.

En la ventana de diálogo Filtro avanzado seleccione dónde desea el listado, cuál es el rango de criterios y a partir de qué celda se debe recibir los resultados.

Nota 1:

En el caso de Excel 2003, si desea extraer lo filtrado hacia otra hoja, entonces debe definir el rango de los datos con un nombre de rango. Si desea, el rango de criterios puede estar otra hoja, sólo que, para mayor facilidad se puede usar también nombre de rango. En la versión 2007 no es necesario que el rango tenga nombre, pero sí que el procedimiento se ejecute estando en la hoja hacia donde se desea el resultado.

Ejemplo

Abra el archivo **Pedidos.xlsx**. En la hoja Pedidos se tiene una lista de pedidos de 830 clientes, atendidos por un grupo de empleados y enviados a distintos destinos. En la hoja Detalles de pedidos, se tiene la lista de productos pedidos por cada cliente. Se desea obtener

a) Un reporte de los pedidos realizados por un cliente en particular.

b)Igualmente se desea obtener un reporte de todos los pedidos atendidos por dos empleados en particular y cuya forma de envío sea Speddy Express.

Solución

Haciendo clic en el cuadro de nombres, apreciamos que los datos de la hoja **Pedidos** tiene por nombre de rango, **Pedidos** y que la otra hoja tiene por nombre, **Detalle**.

Pregunta a)

Insertemos una nueva hoja; que su nombre sea Reporte 1.

Copiemos toda la cabecera (nombres de campo) de la hoja Detalles de pedidos y la pegamos en la primera fila de la hoja Reporte 1

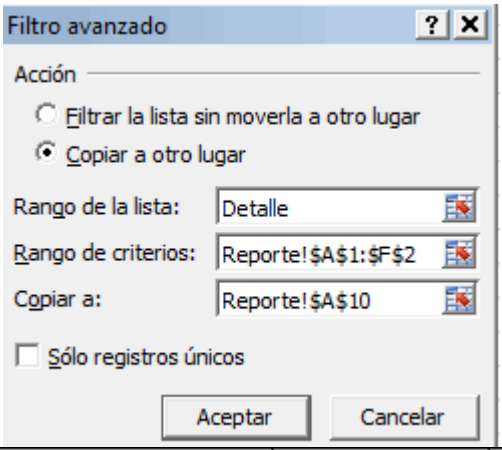
Supongamos que se desea obtener los pedidos del cliente cuyo número de pedido es 11077. Para ello digitamos debajo de Id Pedido, 11077 (en la hoja Reporte)

	A	B	C	D	E	F
1	Id de pedido	Producto	Precio por unidad	Cantidad	Descuento	Monto
2	11077					

Ahora realizaremos el procedimiento del Filtro Avanzado:

Estando en la hoja Reporte, hacemos clic en <Avanzadas> del grupo <Ordenar y filtrar> de la ficha <Datos>.

Completamos la ventana de diálogo que aparece, según se muestra en la siguiente imagen:



Id de pedido	Producto	Precio por unidad	Cantidad	Descuento	Monto
11077	Cerveza tibetana Barley	$ 19.00	1638	20.00%	24897.60
11077	Sirope de regaliz	$ 10.00	2907	0.00%	29070.00
11077	Especias Cajun del chef Anton	$ 22.00	477	0.00%	10494.00
11077	Mermelada de grosellas de la abuela	$ 25.00	896	2.00%	21952.00
11077	Peras secas orgánicas del tío Bob	$ 30.00	158	5.00%	4503.00
11077	Salsa de arándanos Northwoods	$ 40.00	390	10.00%	14040.00
11077	Pez espada	$ 31.00	918	0.00%	28458.00
11077	Queso Manchego La Pastora	$ 38.00	1344	5.00%	48518.40
11077	Algas Konbu	$ 6.00	1716	0.00%	10296.00
11077	Cuajada de judías	$ 23.25	199	3.00%	4487.95
11077	Postre de merengue Pavlova	$ 17.45	1504	3.00%	25457.46
11077	Mermelada de Sir Rodney's	$ 81.00	780	4.00%	60652.80
11077	Pan fino	$ 9.00	1960	0.00%	17640.00

A la derecha se aprecia una parte de dicho reporte.

Pregunta b)

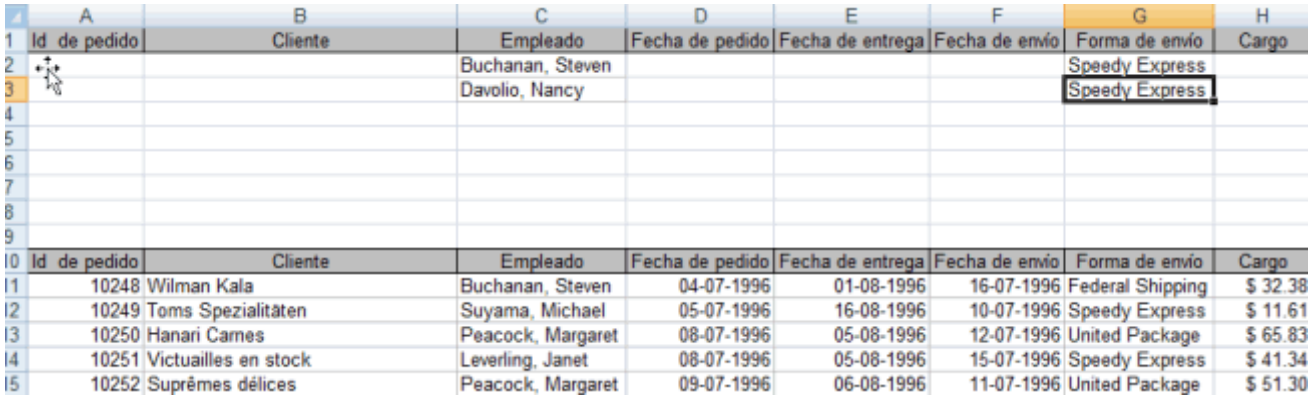
Inserte una nueva hoja y que se nombre sea Reporte 2.

Copiaremos la cabecera de la hoja Pedidos y la pegamos en esta nueva hoja.

Supongamos que se desea obtener un reporte de los pedidos atendidos por Buchanan, Steven y por Davolio, Nancy. Esto significa que debemos ingresar estos nombres debajo de la columna Empleado y "Speddy Express" lo ingresamos debajo de Forma de envío pero repetido, como se muestra en la siguiente imagen:

Empleado	Fecha de pedido	Fecha de entrega	Fecha de envío	Forma de envío
Buchanan, Steven				Speedy Express
Davolio, Nancy				Speedy Express

A continuación realice el mismo procedimiento anterior digitando en Rango de la lista: Pedidos. La siguiente imagen muestra parte de este reporte.



Id de pedido	Cliente	Empleado	Fecha de pedido	Fecha de entrega	Fecha de envío	Forma de envío	Cargo
		Buchanan, Steven				Speedy Express	
		Davolio, Nancy				Speedy Express	
10248	Wilman Kala	Buchanan, Steven	04-07-1996	01-08-1996	16-07-1996	Federal Shipping	$ 32.38
10249	Toms Spezialitäten	Suyama, Michael	05-07-1996	16-08-1996	10-07-1996	Speedy Express	$ 11.61
10250	Hanari Carnes	Peacock, Margaret	08-07-1996	05-08-1996	12-07-1996	United Package	$ 65.83
10251	Victuailles en stock	Leverling, Janet	08-07-1996	05-08-1996	15-07-1996	Speedy Express	$ 41.34
10252	Suprêmes délices	Peacock, Margaret	09-07-1996	06-08-1996	11-07-1996	United Package	$ 51.30

Consolidación

Se puede consolidar rangos de datos que están en una misma hoja, rangos de datos que están en diferentes hojas o igualmente rango de datos que corresponden a libros diferentes.

Nota 1:

Se debe tomar en cuenta que la consolidación se lleva a cabo sobre rangos que tiene la misma estructura, aunque el número de filas puede variar entre un rango y otro. Para los temas de consolidación use el archivo **Ventas anuales.xlsx** y **Ventas diarias.xlsx**.

Nota 2:

Si los rangos tuvieran columnas que no puedan ser consolidadas, puede dejar de incluirlas en la selección y usar las siguientes o trasladarlas al final a fin de que se pueda elegir un rango adecuado. Siempre consolida tomando en cuenta la primera columna del rango.

Consolidación de rangos de una misma hoja

La Figura 2 muestra la secuencia de pasos para realizar una consolidación de rangos dentro de una misma hoja de un libro.

Observación

En la imagen de la Figura 2, se han consolidado cuatro rangos contenidos en la misma hoja. Haciendo clic en el cuadro <Referencia> se han seleccionado cada rango y luego se hizo clic en <Agregar>. Esto para cada rango a ser consolidado..

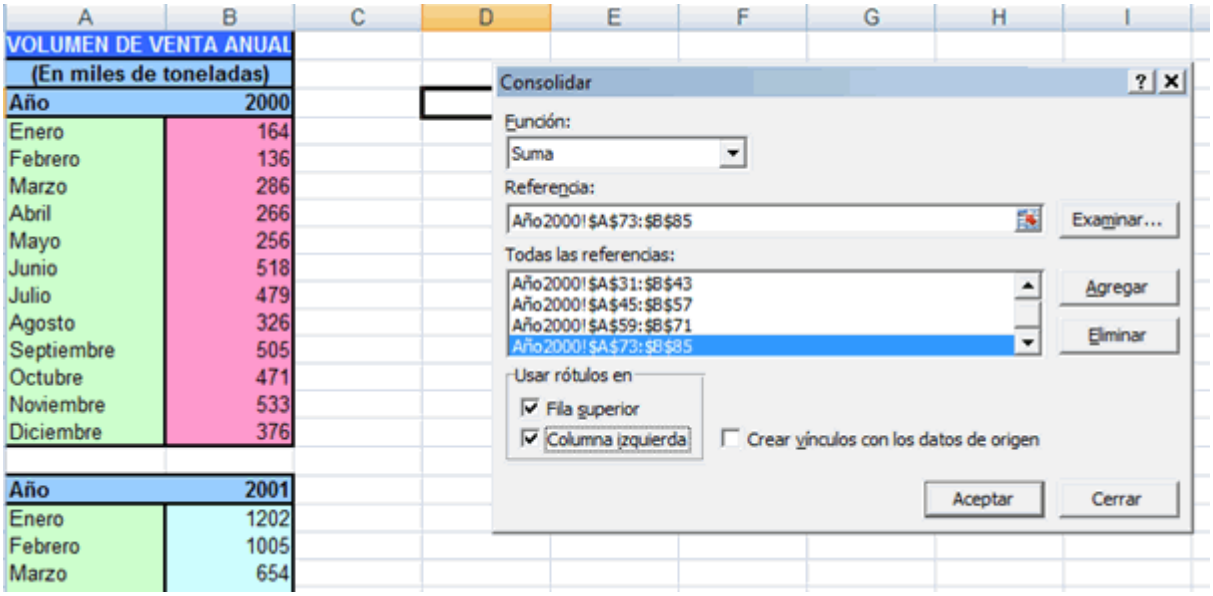


Figura 2

Consolidación de hojas

En este caso, los rangos de consolidación se encuentran en hojas diferentes, aunque no necesariamente deben estar todos en hojas diferentes.

La secuencia de acciones se muestra en la figura 03.

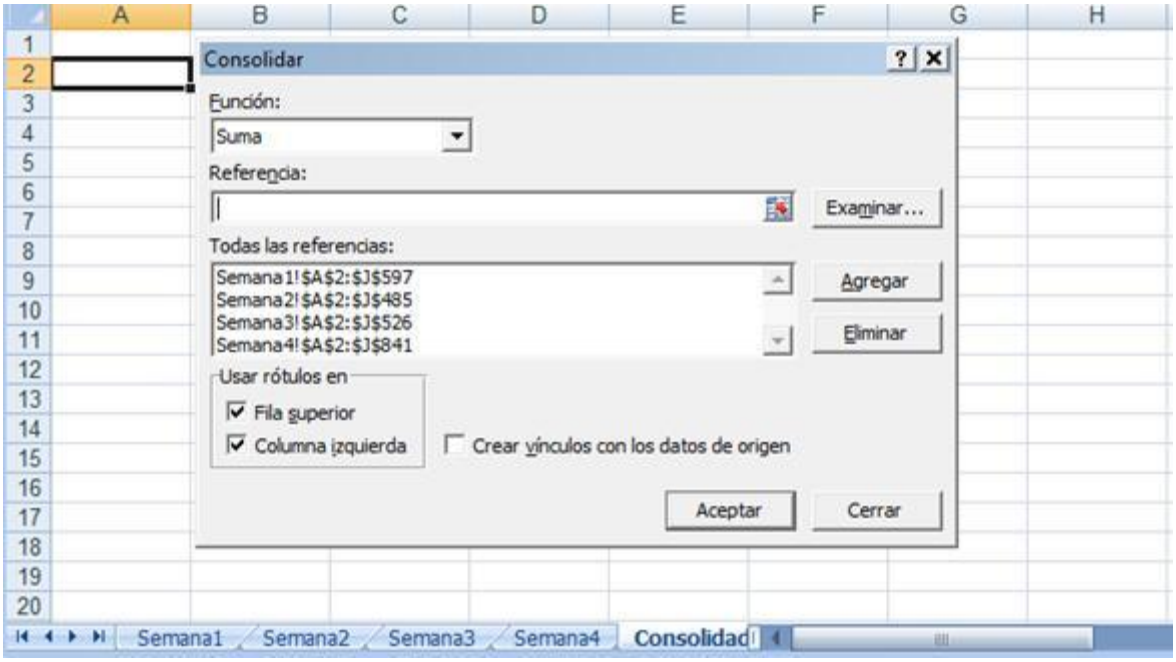


Figura 3

En esta imagen se ha seleccionado el rango correspondiente a la hoja **Semana1** y se procede a <**Agregar**> a la consolidación la **Semana2** y todas las otras semanas.

Nota 3

A fin de llevar a cabo una consolidación interesante, abra el archivo **Proyecciones.xls** y realice la consolidación de las proyecciones de todos los meses.

Nota 4

Recuerde que, si desea una consolidación efectuando una suma, promedio, etc. Sobre la misma columna de consolidación, dicha columna debe tener la misma cabecera en todos los componentes (rangos) de la consolidación. En este último caso, en lugar de tener 2001, 2002, etc, se debiera tener un mismo rótulo.

Consolidación de libros

Igualmente, si los rangos a ser consolidados estuvieran en libros diferentes, también se pueden realizar dicha acción. Para ello es necesario que estén abiertos todos los libros a ser consolidados así como un nuevo libro donde se realizará la misma.

Tablas dinámicas

Una de las herramientas de mayor uso en el Excel es la tabla dinámica. Los datos a ser empleados provienen de una tabla o base de datos en general. Tiene la característica de ser dinámica por cuanto los elementos que conforman su estructura pueden ser modificados (añadiendo o quitando campos) en el instante. Una tabla dinámica está ligada a un gráfico dinámico y cualquier cambio de uno afecta al otro.

Una tabla dinámica puede construirse a partir de los datos contenidos en un libro del Excel, con los datos provenientes de una consulta por ejemplo del MS Access o directamente del MS Query.

Del mismo modo, una tabla dinámica puede ser construida desde un cubo OLAP (On Line Analitical Processing), aunque éste ya tiene diseñado las dimensiones (cada una de las cuales generan de por sí una tabla), podemos insertar en la misma tabla más de una dimensión, enriqueciéndola aún más.

Si los datos que se tienen no están en formato del Excel, no son consultas o no tienen un formato que permita acceder automáticamente desde el Excel, podemos realizar el proceso de conversión mediante el uso de la secuencia: <Datos> - <Obtener datos externos> - <Importar datos>.

Las siguientes figuras muestran la secuencia de acciones necesarias para construir una tabla dinámica. El archivo a ser usado es **Comercial plaza.xlsx**.

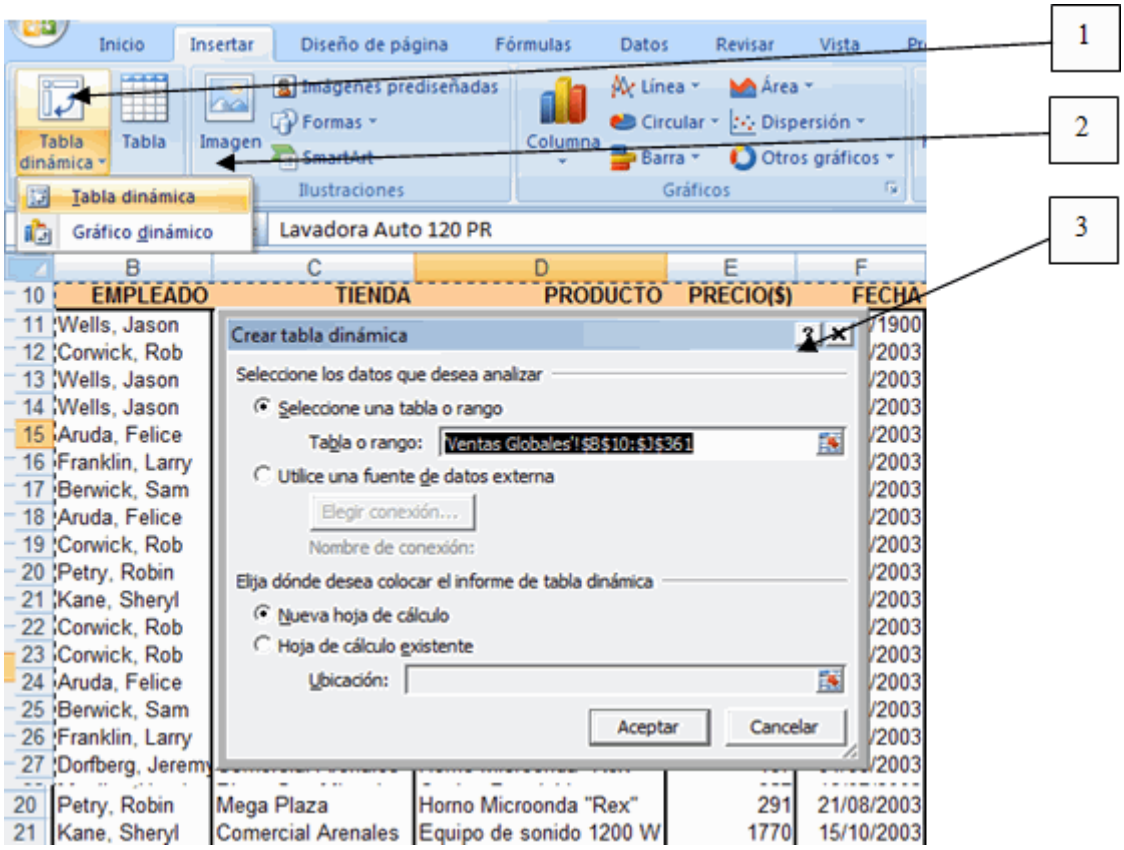


Figura 4

Nota 1:

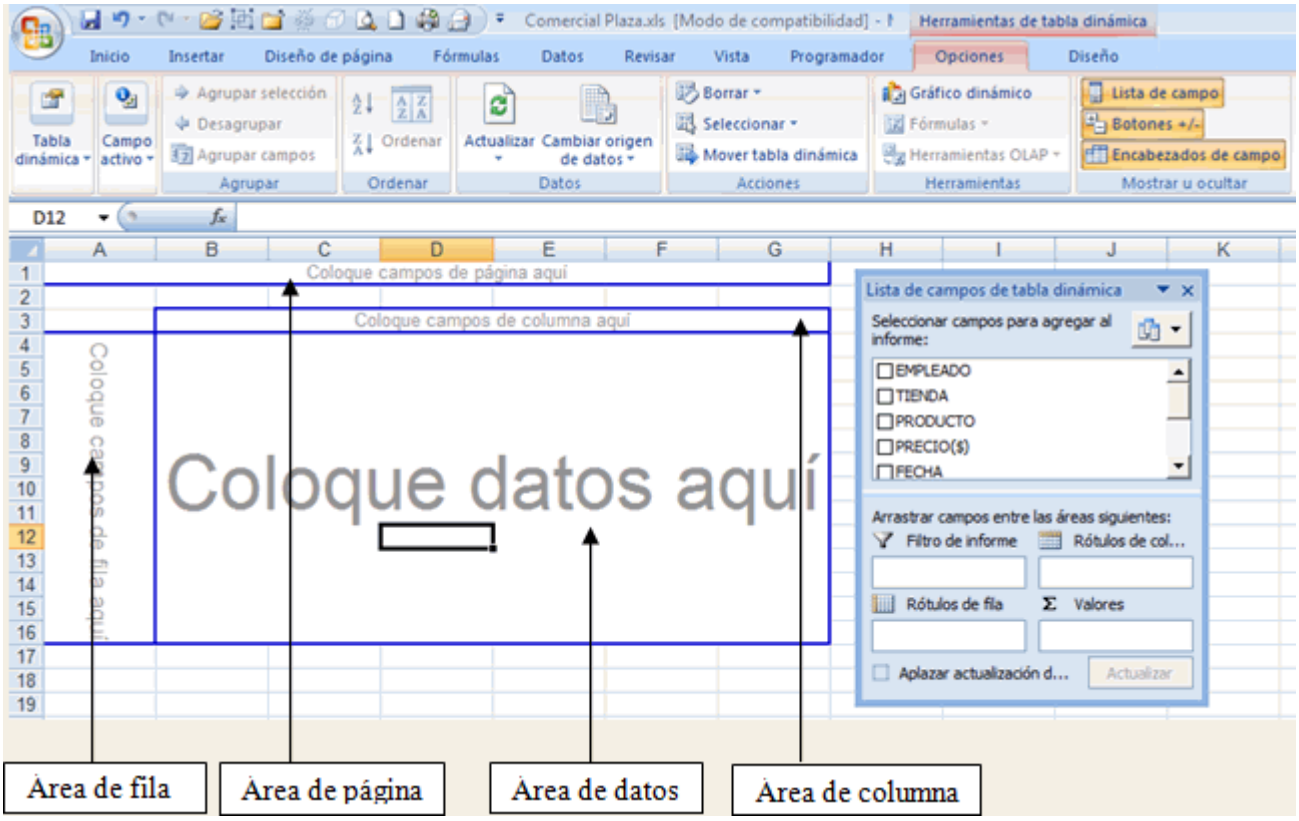
Para obtener una tabla dinámica en Excel 2003 siga el siguiente procedimiento:

Hacer clic en cualquier celda dentro del rango de los datos.

<Datos> - <Informe de Tablas y Gráficos dinámicos> - <Lista o base de datos de Microsoft Excel> - <Siguiente> - Verificar si el rango que se muestra es el indicado - <Siguiente> . En la ventana siguiente seleccionar dónde se desea el resultado y luego hacer clic en <Diseño>. En esta ventana debe seleccionar el o los campos que debe colocar en el área de fila, en el área de columna y en área de datos (a los cuales de preferencia se debe colocar los campos numéricos).

Nota 2:

Como se aprecia en (1), se debe hacer clic en la <Tabla dinámica> de la ficha <Insertar>. El cursor debe estar dentro de los datos a fin de disponer del rango ya seleccionado como se muestra en (3). En esta ventana se debe decidir si la tabla se desea en una nueva hoja o en la hoja donde están los datos. Luego de hacer clic en <Aceptar> se obtiene la siguiente ventana:



Arrastre los campos de la Lista de campos hacia el área donde desea se despliegue los valores de dicho campo. Al área de datos deben ir uno o más campos capaces de ser sumarizados o los campos categóricos con la idea de contar su repitencia.

Nota 3:

En el caso de la versión 2007 es suficiente arrastrar los campos de la lista de campos hacia uno de los cuatro cuadros de texto de la parte inferior de la misma.

Nota 4:

Si desea modificar el diseño de la tabla es suficiente arrastrar el botón y soltarlo fuera de la tabla y colocar en su lugar otro. Puede igualmente quitar todos los botones y cambiar totalmente el diseño inicial.

Nota 5: Cómo crear una tabla dinámica con datos que no tienen formato de Excel.

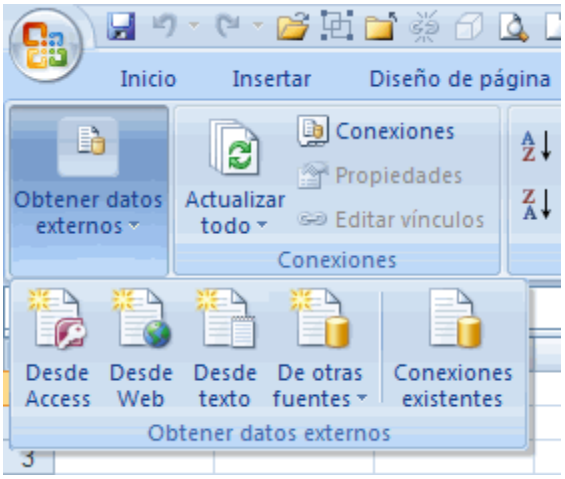
Si los datos no están en el formato del Excel, puede usar previamente la importación de los datos usando la secuencia:

En Excel 2003:

<Datos> - <Obtener datos externos>. Seleccionar la fuente de datos. Por ejemplo, si el archivo es de tipo texto (extensiones TXT, PRN, CSV), haga clic en <Desde texto>.

En Excel 2007:

Al hacer clic en la ficha <Datos> obtendrá las siguientes opciones



Haga clic en <Obtener datos externos> y seleccione la opción correspondiente al formato en el cual se encuentra los datos a ser convertidos al formato del Excel.

En ambos casos y a partir de ello, seguir la secuencia del asistente para importar datos. En ella seleccione la opción que le corresponda y tomando en cuenta el separador de campo. Para ello sería bueno abrir el archivo fuente usando algún programa como un editor de textos como el bloc de notas o algún otro.

Nota 6:

En los casos en los que se desea crear una tabla dinámica con datos de otras fuentes,

En el numeral 3, de la gráfica inicial de tabla dinámica, use la opción <De otras fuentes> de la ficha <Datos>, si desea acceder a archivos de consulta grabados usando el MS Query o para crear cubos OLAP o acceder a uno de ellos.

Análisis de datos

El Excel dispone de una herramienta llamada análisis de datos que nos permite resolver problemas estadísticos que implican el uso de un determinado procedimiento a diferencia de las funciones estadísticas que emiten resultados independientes de un contexto. Entre los procedimientos disponibles en esta herramienta están: Estadística descriptiva, Muestreo aleatorio simple y sistemático, análisis de regresión múltiple, análisis de covarianzas, análisis de correlación múltiple, algunos modelos de prueba de hipótesis, de análisis de varianza de uno y dos factores, etc.

La figura 5 muestra la secuencia de acciones para obtener las estadísticas descriptivas de una serie de datos.

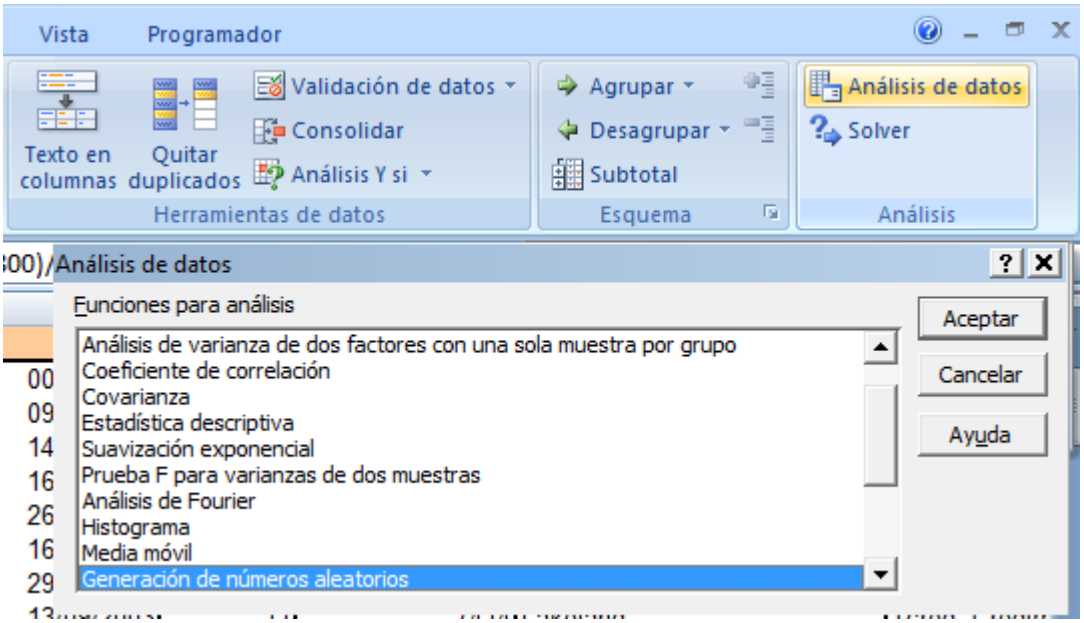


Figura 5

Nota:

El uso de cada una de las opciones de esta herramienta requiere de conocimientos estadísticos tanto descriptivos como inferenciales.

Aplicaciones de las macros

En una consolidación

Procedimiento para grabar una macro:

En Excel 2003:

Puede usar una de las siguientes opciones:

Activar la barra de herramientas de Visual Basic y hacer clic en el botón <Grabar nueva macro>.

Puede usar la secuencia <Herramientas> - <Macro> - <Grabar nueva macro>.

En Excel 2007:

Debe tener activada la ficha del Programador.

Hacer clic en Grabar macro del grupo Código

A continuación y en ambas versiones, debe dar nombre a la macro (sin dejar espacio en blanco); ingresar una tecla para el método abreviado y verificar dónde desea grabar la macro: en el libro actual, en un nuevo libro o en el libro de macros personal.

Ahora ya debe realizar todas las operaciones que desea que haga la macro

Al final (y sin hacer otra cosa) debe detener la grabación de la macro haciendo clic en un pequeño botón que se dispone en Excel 2003 o en el icono que aparece en el lado izquierdo de la barra de estado en Excel 2007.

Ahora sí pasemos a los ejemplos de aplicación de macros.

Ejemplo

Grabe una macro que permita consolidar datos de proyecciones mensuales, cada una de las cuales está en una hoja diferente. Para ello abra el archivo **Proyecciones.xlsx**.

Antes de iniciar la grabación verifique lo siguiente:

Los rangos a ser consolidados tienen la misma estructura en todas las hojas, aunque la cantidad de filas puede no ser la misma

Para mayor facilidad al ingresar los rangos, sería bueno usar nombres de rangos y no rangos mismos. Para ello ya hemos dado nombre a cada uno de ellos.

Se debe tener cuidado de ordenar las hojas ya que el Excel ordena alfabéticamente los rango de consolidación en la lista <Agregar>.

Abra el archivo **Proyecciones.xls**. Ubíquese en la hoja <Análisis de ventas>

El nombre de la macro es MacCons01; el método abreviado es <CTRL> + <k>

Solución

Antes de iniciar la grabación de la macro, daremos nombre a cada uno de los rangos. Puesto que el rango es el mismo, démosle nombre Mes01 a Enero, Mes02 a Febrero, etc.

Durante la grabación de la macro, en el cuadro de referencia, digite el nombre del rango de cada hoja y haga clic en <Agregar>.

A continuación active las opciones de Fila superior y columna izquierda. Luego detenga la grabación.

Para ejecutar la macro debe eliminar los rangos consolidados de la ventana de consolidación.

En una tabla dinámica

Ejemplo

Crear una tabla dinámica básica usando los datos del archivo ***TablaDin01.xls***. Suponer que las hojas son datos copiados desde otros libros provenientes de la zona Norte, Oeste y Este.

Solución

Al abrir este archivo veremos que los datos se encuentran en tres hojas, cada una de las cuales contiene las ventas en las zonas Norte, Oeste y Sur.

El objetivo es disponer de una nueva hoja que contenga los datos de todas las zonas y a partir de ella, se pueda crear la tabla dinámica básica. Llamamos tabla dinámica básica pues la macro creará una con los mínimos elementos; a partir de la cual se puede añadir o remover los campos de las áreas respectivas.

La macro se llamará *TablaDinamica01*. El método abreviado, <CTRL>+<m>.

Secuencia de acciones que debe hacerse al grabar la macro:

Insertar una nueva hoja, darle nombre: Datos

Copiar toda la hoja Norte hacia la hoja Datos, a partir de la celda A1

Copiar sólo los datos de Oeste hacia Datos, a partir de la primera fila vacía

Copiar sólo los datos de Sur hacia Datos, a partir de la primera fila vacía

Ubicarse al interior de la hoja Datos

Usar la secuencia para crear la tabla dinámica diseñando la siguiente estructura.

<table>
<tr><td colspan="3"></td></tr>
<tr><td colspan="2"></td><td>Zona</td></tr>
<tr><td rowspan="2">Tienda</td><td colspan="2"></td></tr>
<tr><td>Forma de Pago</td><td>Suma de Monto</td></tr>
</table>

Detener grabación de la macro

En Filtro avanzado

Otra de las herramientas que podemos automatizar mediante el uso de macros es la de Filtro Avanzado.

La potencia de la macro está en el hecho de modificar los criterios de filtrado no tanto en la ejecución de la macro con los valores iniciales, que de por sí serían bastante rígidos.

Habiendo establecido un valor para el filtrado, podemos modificar dicho valor o agregar otros (en la misma fila, no más columnas). Puesto que la salida será siempre a partir de la celda ingresada durante la grabación de la macro y, estando ocupado dicho rango con una ejecución anterior, podemos grabar nueva macro que permita borrar la cabecera en la salida y volver ejecutar la macro.

Ejemplo 1

Grabar una macro para extraer todos los pedidos atendidos un determinado empleado. Use el archivo **Pedidos.xls**.

Procedimiento:

Abrir el archivo **Pedidos.xls**

Insertar una hoja, colocarla al final y darle nombre *Filtrado*.

Supongamos que se desea extraer los pedidos atendidos por el empleado *Davolio, Nancy* con forma de envío, *Speedy Express*. Se ingresa esos datos.

Iniciamos la grabación de la macro. Noimbre: MacFiltro01; método abreviado: <CTRL> + <o>

Estando en grabación: En Excel 2003: <Datos> - <Filtro> - <Filtro avanzado>

En Excel 2007: Ficha Datos – Clic en <Avanzadas> del grupo <ordenar y filtrar>.

En ambas versiones: <Copiar a otro lugar> - <Rango de la lista: Pedidos> - <Rango de criterios: A1:L2 - <Copiar a: A10> - <Aceptar>

Detener grabación

Grabemos otra macro llamada MacBorrar01; método abreviado: <CTRL> + <b> que seleccione el rango A10:L10 y lo borre; luego detener grabación.

Ahora ejecute la macro MacFiltro01 usando el método abreviado.

Cambie el nombre del empleado por otro de la hoja Pedidos y antes de volver a ejecutar la macro, borre la cabecera de la salida, ejecutando le macro MacBorrar01.

Ahora agregue más criterios de filtrado en la misma línea para reducir la lista de salida; antes de volver a ejecutar la macro debe borrar la salida.

Observación 1

La macro permite extraer los pedidos atendidos por más de un empleado?. No. Si se quisiera que extraiga los pedidos atendidos por dos empleados, deberíamos colocar los nombres uno debajo de otro. Por ejemplo: En C2 colocar Davolio, Nancy y en C3 colocar Buchanan, Steven. El listado obtenido sería los pedidos atendidos por Davolio o Buchanan. Esto no hace la macro pues ella sólo lee la primera fila de criterios. Si sólo pudiéramos variar el rango de A1:L2 al rango A1:L3, habríamos resuelto el problema. Esto implica modificar la macro.

Ejemplo 2

Qué ocurre si después de tener la lista de los pedidos atendidos por un determinado empleado, quisiéramos saber el detalle del pedido de un determinado cliente? Esto significa trabajar con la hoja *Detalle de Pedidos*.

La macro que grabaremos en este ejemplo será ejecutado después de ejecutar la macro del ejemplo anterior.

Procedimiento:

• Ejecutar la macro MacFiltro01

• Copiar el Nro. de pedido (Id. de pedido) de alguno de los pedidos obtenidos al ejecutar la macro anterior hacia la celda A2.

• Iniciar la grabación de una nueva macro. Nombre: MacFiltro02; método abreviado: <CTRL> + <j>.

• En O8 digitar: =BUSCARV(A2,Pedidos,2,0) para extraer nombre de cliente

• En P8 digitar: =BUSCARV(A2,Pedidos,4,0) para extraer fecha de pedido

• Usar <Datos> - <Filtro> - <Filtro avanzado>: Copiar a otro lugar; Rango de lista: Detalle; rango de criterios: A1:A2; rango de salida: N10 - <Aceptar>.

Grabar la macro MacBorrar02 que sólo borre el rango N10:S10.

Ejecutar la macro.

Si se desea el detalle de otro cliente, será suficiente copiar o digitar el Id de pedido del cliente deseado de la salida de la MacFiltro01 y ejecutar la macro MacFiltro02 borrando previamente la salida con MacBorrar02.

Botones de control de formularios para ejecutar marcos

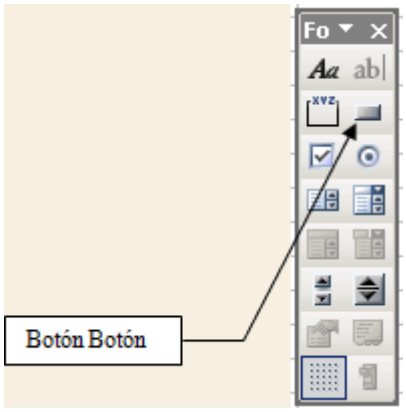
Como habrá podido comprobar, el uso del método abreviado para ejecutar una macro es limitado sea por que no podemos usar cualquier letra ya que inhabilitaría las definidas por omisión, sea por que en un determinado libro podemos tener muchas macros y no sabríamos distinguir una de otra, o por alguna otra razón.

Para evitar esto, el Excel dispone de un conjunto de elementos (botones) de control o de formulario a los cuales podemos asignarle una determinada macro. De esta manera, no tenemos limitación alguna pues podemos tener tantos botones como macros tengamos en un libro.

Estos botones de control son de dos tipos:

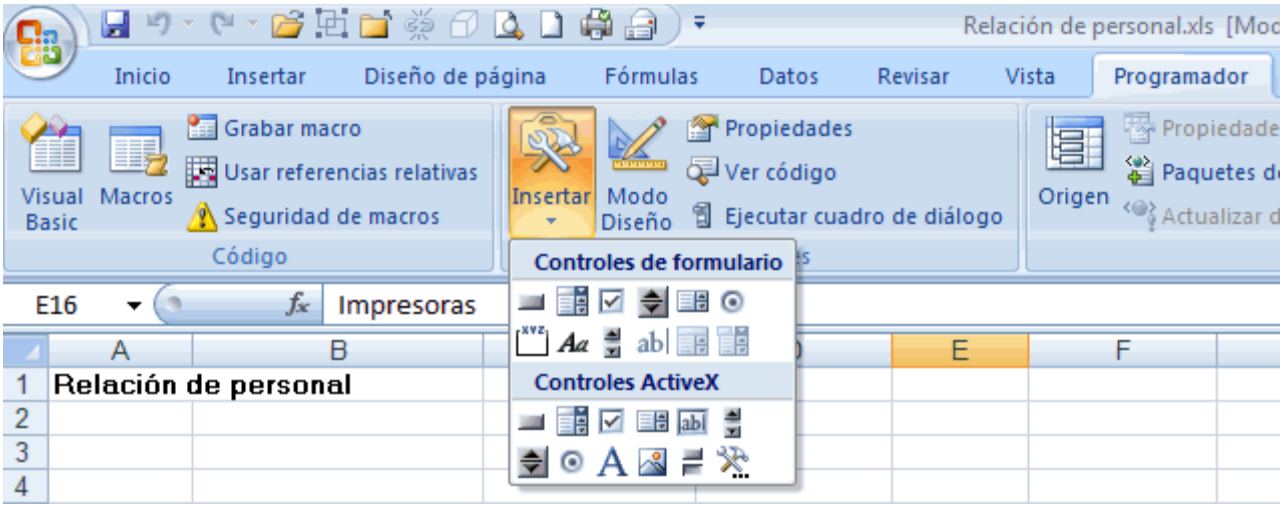
Los llamados "**Controles Activex**" y que conforman la barra de herramientas del Cuadro de controles, y

Los llamados "**Controles de Formulario**" y que conforman la barra de herramientas de Formulario.



Los primeros requieren de una programación (módulo) a la cual se les asigna y el uso de los botones de Formulario son más simples. Estos pueden ser directamente asignados a una macro o pueden ser usado para manipular listas, textos, barra de desplazamiento, casillas de verificación o selección, etc. Entre ellas disponemos del botón llamado Botón Botón o Button.

La imagen anterior corresponde a la barra de herramientas de Formulario, en el caso del Excel 2003. Sin embargo en el caso de la versión 2007, se accede a estos botones haciendo clic en el comando **Insertar** del grupo **Controles** de la ficha del **Programador**, como se muestra en la siguiente imagen:



Procedimiento para asignar un botón a una macro:

En el caso del Excel 2003:

Paso 1: Active la barra de herramientas de Formulario usando la siguiente secuencia:

<Ver> - <Barra de Herramientas> - <Formulario>

En el caso del Excel 2007

Paso 1: <Programador> - Desplegar la lista de <Insertar> del grupo Controles.

Paso 2: Haga clic en el botón "Botón" y trace un rectángulo (dibuje) en una parte de la hoja donde desee colocar el botón

Paso 3: Si al soltar el botón del mouse, no se abre una ventana, haga clic con el botón derecho del mismo y seleccione la opción <Asignar macro>. Seleccione el nombre de la macro al que desea asignar el botón y luego haga clic en <Aceptar>.

Para ejecutar la macro es suficiente hacer clic en dicho botón.

Ejemplo 1

Abra el archivo **Pedidos.xls**. Vaya a la hoja **Filtrado**. Vamos a obtener un listado de todos los pedidos atendidos por **Davolio, Nancy**. Para ello digite dicho nombre en C2.

Ahora, usando el procedimiento, descrito líneas arriba, inserte un botón en G7 y cuando tenga que ingresar el nombre de la macro, seleccione **MacFiltro01**. Luego haga clic en Aceptar.

Vamos a cambiar de nombre al botón. Haga clic con el botón derecho del mouse. Como verá, se despliega un menú contextual que no nos interesa. Presione la tecla <Esc> y digite **Filtrar Pedidos**.

Ahora ya puede identificar claramente para qué sirve el botón y para ejecutarlo es suficiente hacer clic en él.

Ejemplo 2

Usando el mismo archivo anterior, inserte otro botón botón en G8. Asigne este botón a la macro MacBorrar01. Haga que su nombre sea Borrar cabecera.

Unidad 2. Programación orientada a objetos

La programación orientada a objetos (POO) es una forma de programación en computadoras que tiene surge los años 70 pero tiene un desarrollo sorprendente los años 90 al utilizarlo 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.

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 **Automóvil**.

Un automóvil es un objeto bastante pesado que tiene un conjunto de propiedades como su identificación (placa), 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. Tienen de la capacidad de ser activados para poner en acción sus funcionalidades; es decir, disponemos de un procedimiento para ponerlo en marcha, avanzar en retroceso, detenerlo, voltear a la izquierda o derecha; es decir, mediante un conjunto de métodos podemos darle uso al objeto automóvil.

En la POO el objeto es el automóvil; las **propiedades** de este objeto son sus características y los **métodos** lo constituyen las funcionalidades o procedimientos con los cuales hacemos uso del objeto y modificamos su estado o contenido.

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

Son 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:

MiAuto.Color = Verde

donde el punto recibe el nombre de operador.

Aquí, MiAuto es una instancia del objeto Automóvil; 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 omisión 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**.

Método

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 automóvil 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:

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**.

Unidad 3. Lenguaje Visual Basic de Aplicaciones (VBA)

Antes de empezar el estudio de las instrucciones del Visual Basic para Aplicaciones, daremos una breve explicación del Editor de Visual Basic y los elementos dentro de su entorno.

El Editor del Visual Basic

El Editor de Visual Basic es un programa cuya ventana principal le permite realizar una serie de acciones sobre su macro, las propiedades de su libro y hojas, así como el de crear nuevos módulos (procedimientos) y formularios. Aquí podemos modificar las macros y potenciarlas dándole la interactividad que no posee. Para usar el editor, haga uso de la siguiente secuencia:

<Herramientas> - <Macro> - <Editor de Visual Basic> o también usando <ALT> + <F11>.

Si se desea editar o abrir una macro en particular, seleccione:

<Herramientas> - <Macro> - <Macros> Seleccione la macro - <Modificar>.

Puede usar también, <ALT> +<F8>, luego seleccionar la macro y hacer clic en <Modificar>.

El Editor y la programación orientada a objetos

Todo lo que se puede hacer en el editor, está relacionada con programas. La filosofía de programación usada en este editor, es el de la Programación Orientada a Objetos (POO), la que nos permitirá hacer modificaciones sobre la programación subyacente a una macro y crear nuestros propios programas. Un programa es, en principio, una secuencia de pasos o instrucciones que escribimos para resolver un problema en particular.

En un lenguaje de programación orientada a objetos, las instrucciones se basan en variables, constantes, objetos, propiedades de los objetos, etc., pertenecientes a un ámbito, a un entorno. Todos estos elementos, sean sus componentes o características, como sus relaciones entre otros objetos, están agrupados o "encapsulados" en los objetos. Los objetos en Excel están constituidos por Celdas, Rangos, Hojas, Libros, Gráficos, Tablas, archivos externos, etc.

Los objetos, refiriéndonos a Excel, poseen cuatro características empleados en la codificación de un programa: **Propiedades** de los objetos, **Métodos** para manipularlos,

Eventos que causan la manipulación de los mismos y, **Colecciones** o Clases a las que pertenecen los objetos.

Propiedades

Las propiedades son las características, atributos, formas o aspectos del objeto, a las que se hace referencia mediante el uso de variables. Una propiedad de objeto común de Excel es su **Nombre**, que nos permitirá usarlos en las diferentes usos que hagamos del objeto. De manera que una celda, un rango de celdas, una hoja, libro, gráfico o tabla en Excel, tendrá un nombre con el cual lo identificaremos.

Para hacer referencia a las propiedades mediante los programas en VBA, se usa la sintaxis:

NombreDelObjeto**.**Propiedad

Ejemplo:

Range("A5").**Name** Aquí se hace referencia al nombre del objeto Range de la celda A5.

Métodos

Un método es una acción, un procedimiento (un hilo), que tiene efectos sobre un objeto.

Ejemplo:

Range("A5").Select Aquí se ejecuta el método Select, que permite activar la celda A5 y ponerla en disponible para cualquier otra operación.

Range("B2:G4").Formato En este caso, se ejecuta el método Formato sobre el rango B2:G4. Se supone que el método Formato ya está definido previamente.

Eventos

Un evento es el resultado de una acción, es la forma cómo queda el objeto después de alguna acción sobre él. Por lo general estas acciones son producidas por los métodos que actúan sobre el objeto.

En la versión 2003 del Excel, podemos tener algunos eventos como:

Se seleccionó una celda o rango

Se seleccionó una hoja (haciendo clic en la pestaña o etiqueta de la misma)

Se ha abierto o cerrado un libro.

Se ha activado o desactivado una hoja de cálculo.

Se han calculado de nuevo las fórmulas de una hoja de cálculo.

Se ha seguido un hipervínculo.

Excel incluye varios controladores de eventos, o rutinas de código, que controlan acciones determinadas. Cuando ocurre una de dichas acciones, y ha comunicado a Excel qué desea que haga cuando ocurra el evento, Excel ejecuta el código del controlador de eventos. Por ejemplo, si después de crear un nuevo libro desea que Excel muestre todos los libros abiertos como un conjunto de ventanas en cascada, puede crear el siguiente controlador de eventos:

```
Private Sub App_NewWorkbook(ByVal Wb As Workbook)

   Application.Windows.Arrange xlArrangeStyleCascade

End Sub
```

Nota

No se preocupe si no está seguro de qué hace cada uno de los elementos de la rutina del controlador de eventos. Por ahora, concéntrese en la línea del medio del código, que comunica a Excel que organice las ventanas utilizando el estilo de cascada. Puede obtener el mismo resultado utilizando los comandos de menú de Excel (menú **Ventana**, comando **Organizar**, sub comando **Cascada**). Sin embargo, si se trata de una acción que desea que se ejecute cada vez que ocurra un evento determinado, puede utilizar el lenguaje VBA para que se ejecute automáticamente y así ahorrarse el trabajo.

Colecciones

El Una colección es un grupo o conjunto de objetos contenidos en otro objeto cuyas propiedades son comunes a los objetos componentes.

Puesto que un libro contiene una o más hojas de cálculo, podemos decir que un libro es una colección de hojas de cálculo. Así las cosas, podemos ejecutar algún método sobre esta colección a fin de realizar la misma acción sobre todas ellas. Por ejemplo, seleccionar tres hojas y darle un determinado formato.

En el siguiente ejemplo, se ha programado la opción de imprimir la ruta donde se encuentra el archivo. Como se puede ver (o lo podrá comprender cuando desarrollemos la instrucción FOR …NEXT), la operación de añadir la dirección se realiza sobre tres hojas del libro.

```
For i = 1 to 3

   Worksheets(i).PageSetup.RightFooter = Path

Next i
```

En el siguiente ejemplo se realiza sobre la colección de hojas del libro.

```
For Each Wksht in Worksheets

    Wksht.PageSetup.RightFooter = Path

Next Wksht
```

En vez de incrementar el valor en un bucle For…Next, el bucle For Each…Next busca simplemente el siguiente elemento de la colección **Worksheets** (hojas de cálculo) y se detiene cuando no encuentra uno.

Ventanas en el Editor de Visual Basic

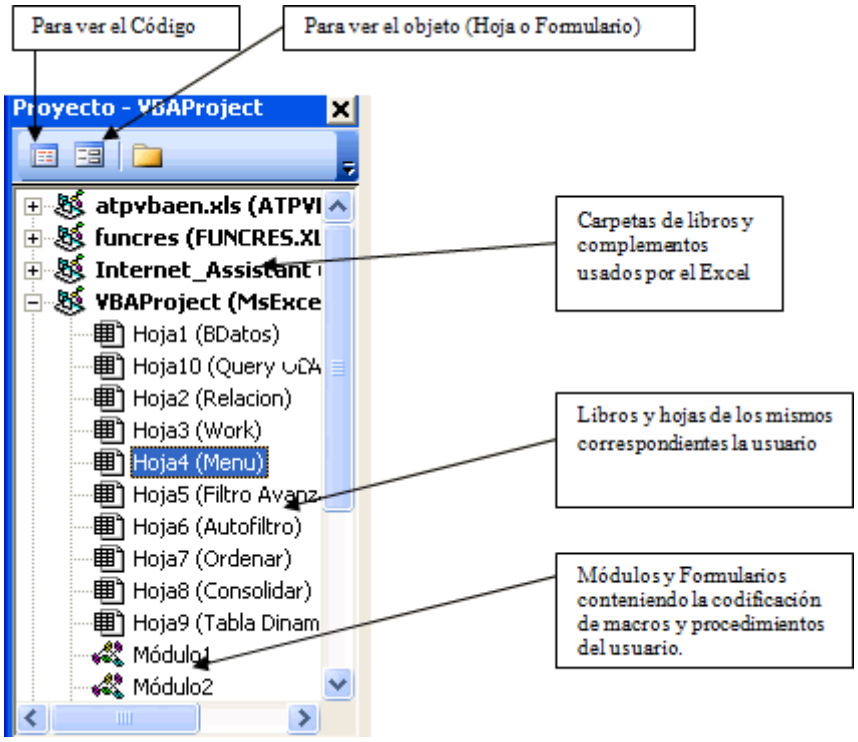
El Editor de Visual Basic muestra distinta información en distintas ventanas. Estas ventanas pueden contener a uno o más módulos (macros) o pueden hacer referencia a las propiedades de los objetos contenidos en la hoja o libro.

Estas son: La ventana Explorador del proyecto, de Propiedades y de Código.

Ventana del Explorador del proyecto

Cuando se abre el Editor de Visual Basic directamente, se puede utilizar la ventana Explorador del proyecto para seleccionar la macro en la que se desea trabajar. El Explorador del proyecto muestra todos los proyectos en términos de carpetas, en vista

de árbol. En ella se muestran, además de los libros que el usuario abre, todos los libros y complementos que el Excel abra en el momento de ejecutar el Excel.



Un módulo puede contener las instrucciones de una o varias macros. El contenido de los libros y complementos del Excel, no se pueden visualizar, excepto si se supiera su contraseña.

Ventana Propiedades

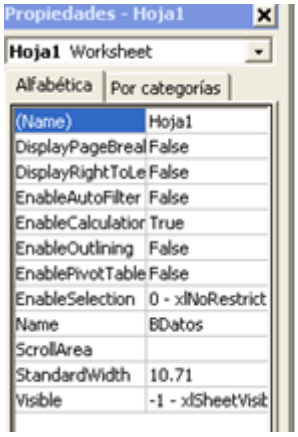
Generalmente debajo de la ventana del Explorador de proyectos, está la ventana de **Propiedades**, que se utiliza para examinar y modificar las distintas propiedades asociadas al objeto seleccionado. La única propiedad que suele estar disponible para los módulos es el **nombre**. Las hojas de cálculo tienen propiedades adicionales que se pueden modificar, como **StandardWidth** y **DisplayPageBreaks,** entre otras.

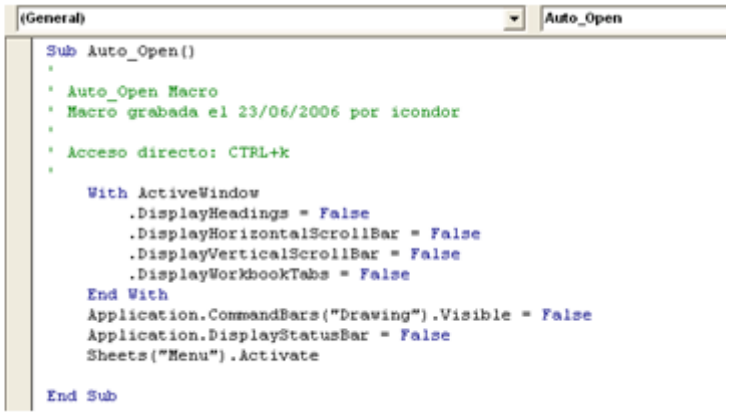
Para modificar las propiedades de un libro u hoja, debemos seleccionarlo primero y luego hacer clic en la propiedad deseada y luego elegir entre las opciones disponibles. Por ejemplo si se desea ocultar la hoja cuyo nombre es Hoja1, selecciónela primero, luego en la ventana propiedades, haga clic en la propiedad <Visible> y seleccione la opción <xlSheetHidden>

Ventana de Código

La ventana de Código es la ventana que dispone de mayor tamaño en el Editor e incluye en el lado superior, dos cuadros de lista desplegables. El cuadro de lista del lado izquierdo (cuadro de Objeto) se utiliza para seleccionar el objeto en el que se quiere trabajar. Cuando se trabaja sólo con código, el cuadro muestra el objeto **General**

predeterminado. El cuadro de lista de la derecha (cuadro de Procedimiento) se utiliza para seleccionar macros individuales del módulo actual. Según se agreguen o eliminen macros en el módulo, se agregarán y eliminarán en el cuadro Procedimiento.





Variables

Puesto que todo lo que se realiza en el computador, debe ser guardado en la memoria, cada uno de los datos deben tener un lugar en ella a donde guardarlos. Para ello es necesario el uso de las variables. Mediante el uso de las variables y también constantes, podemos almacenar cada uno de los datos o constantes en una localidad de la memoria.

Esto implica que, para acceder a dichas localidades, es necesario el uso de las variables. Estas tienen un nombre. El nombre puede estar formado por una letra, seguido de uno o más caracteres literales o numéricos. Estos nombres son únicos; es decir, ninguna otra variable puede tener el mismo nombre. Naturalmente, la variable se puede usar en diferentes ámbitos o secciones; en cada una de ellas, el mismo nombre de variable puede ser usado para propósitos distintos; sin embargo, el nombre de una variable también puede ser definido para ser válido en todos los ámbitos.

Nota:

El nombre de una variable puede estar escrito en minúsculas, mayúsculas o combinarlas a fin de darle cierto sentido respecto a su contenido.

Sentencia DIM

Para declarar una variable se usa la sentencia DIM. Esta declaración se coloca al principio de las secciones o módulos.

Sintaxis:

DIM NombreDeVariable **As** TipoDeDatos

DIM NombreVar1, NombreVar2, … **As** TipoDeDatos

Tipos de variable

La tabla siguiente muestra los tipos de datos compatibles, incluyendo el tamaño de almacenamiento en memoria y el intervalo en el cual pueden variar.

Tipo de datos	Tamaño de almacenamiento	Intervalo
Byte	Entero de un byte	0 a 255
Bolean	Lógico de dos bytes	**True** o **False**
Integer	Entero de dos bytes	-32,768 a 32,767
Long (entero largo)	Entero largo de 4 bytes	-2,147,483,648 a 2,147,483,647
Single (coma flotante/ precisión simple)	Decimal de 4 bytes	-3,402823E38 a –1,401298E-45 para valores negativos; 1,401298E-45 a 3,402823E38 para valores positivos
Double (coma flotante/ precisión doble)	Decimal de 8 bytes	-1.79769313486231E308 a -4,94065645841247E-324 para valores negativos; 4,94065645841247E-324 a 1,79769313486232E308 para valores positivos
Currency (entero a escala)	Entero de 8 bytes	-922.337.203.685.477,5808 a 922.337.203.685.477,5807
Decimal	Números reales con decimales hasta de 28 dígitos (14 bytes)	+/-79.228.162.514.264.337.593.543.950.335 sin punto decimal; +/-7,9228162514264337593543950335 con 28 posiciones a la derecha del signo decimal; el número más pequeño distinto de cero es +/-0,0000000000000000000000000001
Date	Usado para fechas de 8 bytes	1 de enero de 100 a 31 de diciembre de 9999
Object	4 bytes	Cualquier referencia a tipo **Object**
String (longitud variable)	Usado para datos de texto de 10 bytes	Desde 0 a 2.000 millones
String (longitud fija)	Como el anterior pero de longitud fija	Desde 1 a 65.400 aproximadamente
Variant (con números)	Usado como tipo de dato genérico de 16 bytes	Cualquier valor numérico hasta el intervalo de un tipo **Double**
Variant (con caracteres)	22 bytes + longitud de la cadena	El mismo intervalo que para un tipo **String** de longitud variable
Definido por el usuario (utilizando **Type**)	Número requerido por los elementos	El intervalo de cada elemento es el mismo que el intervalo de su tipo de datos.

Ejemplos

DIM A, B As Integer	A y B serán usados como enteros

DIM Xtot, Zdat As Double	Xtot y Zdat serán usados como variables reales
DIM XTitulo, Nom01, Nom02 As String	Usados para contener datos literales
DIM N, Epsi As Variant	Puede contener datos numéricos o literales

Clases de variables

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

Variables Locales

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. Se pueden usar en otros módulos o procedimientos pero sus características nada tienen que hacer con la declaración dada en otro módulo o procedimiento. Para declararlas se debe usar la sentencia **DIM**.

Por ejemplo:

```
Sub procedimiento()

    Dim nDat        As Integer

    Dim Cadena      As String

    Sentencias ....

End Sub
```

Variables Públicas

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; no deben definirse en los módulos correspondientes a las hojas del Libro de Trabajo ni en los módulos correspondientes a UserForms.

Variables Estáticas

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:

```
Sub Procedimiento()

    Static nDat As Integer

    [ Sentencias ...]

End Sub
```

Option Explicit

La sentencia *Option Explicit* permite que el programa se detenga cada vez que VBA encuentre una variable que no ha sido definida. Esto es de gran utilidad cuando se usan muchas variables ya que nos permite identificar rápidamente errores o uso no deseado en el nombre de la variable. Esta sentencia se debe escribir al comienzo del módulo.

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

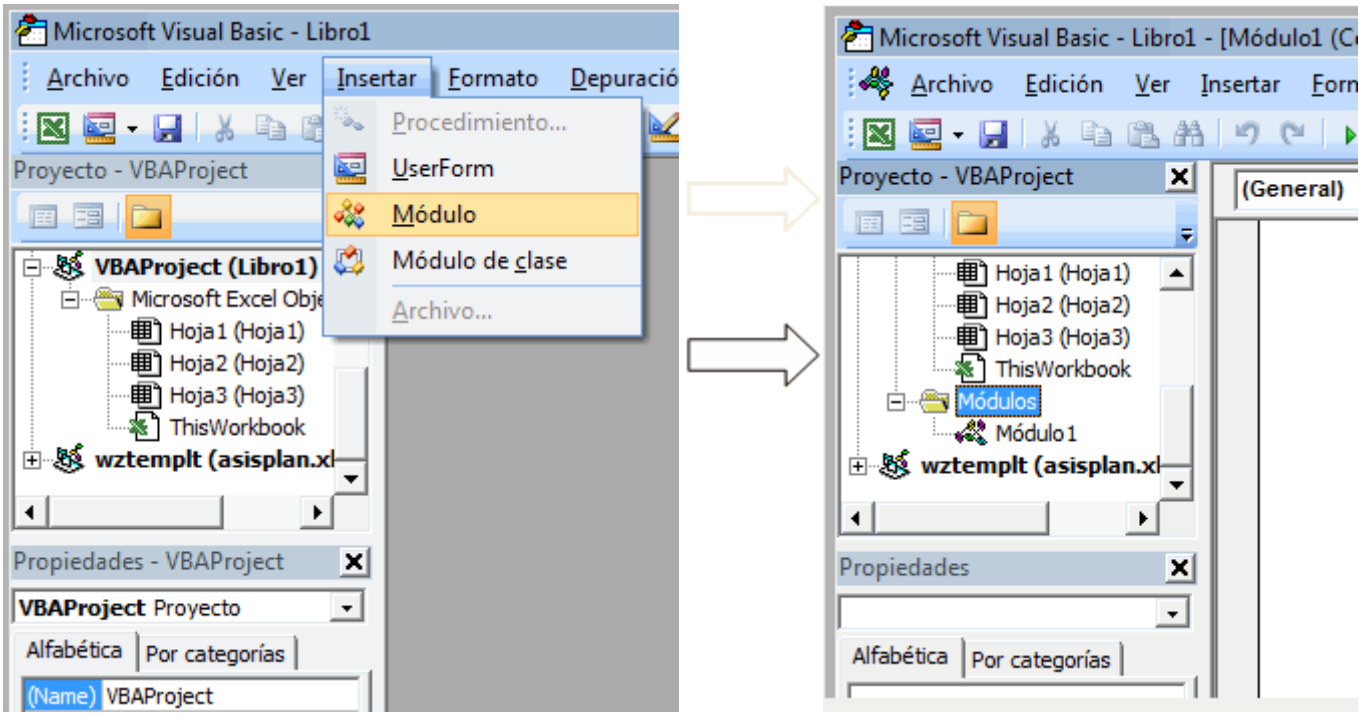
Esta sentencia debe incluirse en un módulo antes del primer procedimiento.

Para definir constantes Locales, basta definirlas a través de la sentencia *Const* dentro de un procedimiento o función.

Módulos

Un módulo es un ambiente, es un entorno de trabajo compuesto por sentencias de declaración de variables y por uno o más procedimientos.

Estando en el Editor del Visual Basic, para disponer de un Módulo, se debe usar la siguiente secuencia de comandos: <Insertar> - <Módulos>, como se indica en la siguiente imagen.



Luego de esta acción, se podrá observar, en la ventana de proyectos y dentro del libro en uso, una carpeta con el nombre Módulo. Dentro de ella se insertarán todos los procedimientos que conforman dicho módulo.

Procedimientos

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

Procedimiento Privado

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

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

[sentencias]

End Sub

Procedimiento Público

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

Procedimiento Estático

Para que las variables de un procedimiento se conserven una vez terminada su ejecución, éste debe definirse como Estático:

Static 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, Public y Static son opcionales. Es importante mencionar que al definir un procedimiento sin ninguna de las sentencias anteriores, por defecto éste se define como Público, es decir:

Sub Procedimiento(argumento1,argumento2,.....)

[Sentencias]

End Sub

Es equivalente a :

Public Sub Procedimiento(argumento1,argumento2,.....)

[Sentencias]

End Sub

Existe una instrucción que permite terminar la ejecución de un procedimiento, ésta es la instrucción *Exit Sub*. Por ejemplo, si tenemos el siguiente procedimiento :

Sub Procedimiento(argumento1,argumento2,.....)

[Sentencias]

Exit Sub

[Sentencias]

End Sub

Llamar a un procedimiento desde otro

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, por ejemplo:

Sub Procedimiento1(argumento1,argumento2,.....)

[Sentencias]

indice= .....

Call Proced2(indice)

[Sentencias]

End Sub

En este caso, el procedimiento *Procedimiento1* llama al procedimiento *Procedimiento2* al cual se le debe pasar el argumento *indice*. Es recomendable utilizar siempre la sentencia *Call* para llamar a otro procedimiento, aunque a éste no se le tengan que pasar argumentos, esto permite identificar más fácilmente las llamadas a otros procedimientos.

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
```

Asignación de valores o expresiones

La asignación de valores en el VBA es muy simple:

VariableReceptora = ValorAsignado / ExpresionAsignada

ObjetoReceptor = ValorAsignado / ExpresionAsignada

El "ValorAsignado" o el resultado de ejecutar "ExpresiónAsignada" se almacena en la localidad de memoria definida como "VariableReceptora" u "ObjetoReceptor".

Ejemplo 1

```
XTot = 120

SumaTot = SumaTot + Num01

XTitulo = "Cuadro de ventas consolidadas"
```

Ejemplo 2

X1 = "Hola Mundo !!!"

X2 = " Buenos días …."

Salida = X1 + X2 + "Hasta luego …"

Ejemplo 3

Tasa = 0.19

Cantidad = 120

PrecioUnit = 25.80

MontoNeto = Cantidad * PrecioUnit – Cantidad * PrecioUnit * Tasa

Nota 1:

En los dos ejemplos anteriores, algunas variables reciben un valor dentro del programa. Toda vez que se ejecute el programa, dichas variables siempre usarán los mismos valores y, como tal, aquellas que dependen de ellas también contendrán los mismos resultados. Para que dichas variables tomen otros valores, es necesario ingresarlos desde otros programas o procedimientos, o desde el teclado.

Nota 2:

En todas las sintaxis de instrucciones que en adelante, se tenga, el uso de corchetes "[…]" nos indicará que su contenido es opcional.

Ingreso de datos. Emisión de resultados

Ingreso de datos: Método InputBox(…)

En VBA los datos se ingresan usando el método InputBox(…) el cual abre una ventana de diálogo. La sintaxis de este método es:

expresión.**InputBox(*Prompt*[,*Title*][,*Default*][,XPos*][,*YPox*] [,*HelpFile*] [,Context*])**

donde

expression Es el nombre de un determinado objeto Application.

Si no se usa, asume Aplication.

Prompt Cadena de caracteres, requerido. Es usado como información al usuario sobre lo que se le pide que ingrese.

Title Cadena de caracteres de tipo Variant, opcional. Es usado para colocarlo en la barra de titulo de la ventana de diálogo.

Default Es opcional. Es el valor que se asigna a la variable receptora, si no se digita ningún dato.

XPos Es la posición horizontal (en pixels) a partir de la cual se visualizará la ventana

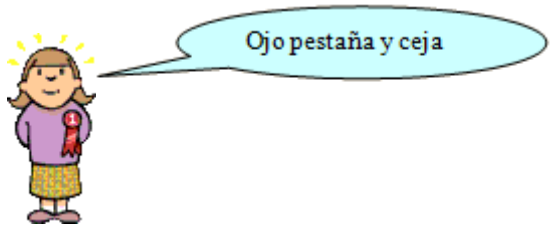
YPos Es la posición vertical (en pixels)a partir de la cual se visualizará la ventana.

HelpFile Es el nombre del archivo de ayuda.

Context Es la posición o ubicación dentro del tema de la ayuda, relativo a la instrucción.

Nota:

Todos los ejemplos de aquí en adelante, pueden ser probados copiando hacia la ventana de código, correspondiente a un módulo. Si no hubiera ningún módulo, haga clic en el comando <Insertar>, luego en <Modulo>.



Nota:

Para ejecutar un procedimiento contenido en un módulo, haga que el cursor se encuentre al interior del procedimiento a ser ejecutado; luego haga clic en  . Para detener la ejecución de un procedimiento que ha fallado, haga clic en  .

Ejemplo 4

En este ejemplo se pide ingresar un número y que si no se digita, la variable Num recibirá el valor que se asume, 120. La ventana de diálogo tendrá por título "Ingreso de datos", se desplegará a partir de las coordenadas (4830, 2210) de la pantalla (en

pixels). En la siguiente instrucción no se coloca un título, si no se digita una expresión, la variable texto contendrá "Hola Perú" y cambiará de posición.

```
Sub datos()

Num = InputBox("Ingresa un número", "Ingreso de datos", 120, 4830, 2210)

Texto = InputBox("Hola Perú",  , 5, 1200, 4800)

End Sub
```

Emisión de resultados: Función MsgBox(..)

Para poder emitir o imprimir en pantalla el contenido de una variable o el resultado de un cálculo, se usa la función MsgBox (…), el cual abre una ventana en cual se visualizará todos los resultados emitidos. La sintaxis de este método es el siguiente:

MsgBox (Prompt [,Botones] [,Title] [,HelpFile] [Context])

donde

Prompt Cadena de caracteres de hasta 1024 bytes. Si ocupa más de una línea se puede separar usando el carácter de retorno de carro CHR(13), o un carácter de avance de línea CHR(10) o una combinación de los dos: CHR(13)+CHR(10).

Botones Permite mostrar u determinado tipo d botón a través de un icono. La siguiente tabla muestra la lista de las variables internas usadas para mostrar el icono.

Constante	Valor	Descripción
VbOKOnly	0	Muestra solamente el botón **Aceptar**.
VbOKCancel	1	Muestra los botones **Aceptar** y **Cancelar**.
VbAbortRetryIgnore	2	Muestra los botones **Anular**, **Reintentar** e **Ignorar**.
VbYesNoCancel	3	Muestra los botones **Sí**, **No** y **Cancelar**.
VbYesNo	4	Muestra los botones **Sí** y **No**.
VbRetryCancel	5	Muestra los botones **Reintentar** y **Cancelar**.
VbCritical	16	Muestra el icono de **mensaje crítico**.
VbQuestion	32	Muestra el icono de **pregunta de advertencia**.
VbExclamation	48	Muestra el icono de **mensaje de advertencia**.
VbInformation	64	Muestra el icono de **mensaje de información**.

El siguiente procedimiento permite ingresar la cantidad y el precio de compra de un producto y luego calcular el monto neto, después de aplicar el impuesto del IGV.

Ejemplo 5

```
Sub Ventas01
'

Dim PrUnit, Neto As Double
Dim Cantidad As Integer
Cantidad = InputBox ("Ingrese la cantidad")
PrUnit = InputBox("Ingrese el precio unitario")
IGV  = InputBox("Valor del IGV", ,0.18)
Neto = Cantidad*PrUnit-Cantidad*PrUnit*IGV
MsgBox("El monto neto es: "+Chr(13)+Chr(10)+Chr(13)+Chr(10)&Neto)
End Sub
```

El siguiente ejemplo muestra el uso de los botones para mostrar sus respectivos iconos

Ejemplo 6

```
Sub Botones
Dim Msg, ComboBotones, Título, Ayuda, Ctxt, Respuesta, MiCadena
Msg = "¿Desea continuar?"
ComboBotones = vbYesNo + vbCritical + vbDefaultButton2    ' Define los botones.
Título = "Prueba de la función MsgBox con botones"       ' Define el título.
Ayuda = "DEMO.HLP"                                       ' Define el archivo de ayuda.
Ctxt = 1000                                               ' Define el tema
                                                          ' el contexto
```

```
Res = MsgBox(Msg, ComboBotones, Título, Ayuda, Ctxt)      ' Muestra el mensaje.
```

```
End Sub
```

Sentencia condicional: If … End If; Select Case …. End Select

Sentencia IF … ELSE … END IF

Sintaxis

```
If ExpresionCondicional Then
  [ Sentencias 1 ]
Else
  [ Sentencias 2 ]
End If
```

Permite ejecutar el grupo de sentencias 1 siempre que la evaluación de "ExpresiónCondicional" de cómo resultado Verdadero; en caso contrario se ejecuta el grupo de sentencias 2.

Ejemplo 7

En el siguiente procedimiento, se imprime en pantalla el cuadrado o el cubo del valor ingresado por teclado, según que éste sea un valor menor o mayor o igual a 10.

```
Sub Potencia
x = InputBox("Ingresa un número entre 0 y 500")
If x < 10 Then
  ValorCalc =x^2
Else
  ValorCalc = x^3
End If
MsgBox(ValorCalc)
```

```
End Sub
```

Nota:

Observe que en este procedimiento no hemos declarado ninguna variable. Esto es correcto también; sin embargo en un procedimiento formal y bastante grande, se recomienda hacerlo.

Nota

La siguiente tabla muestra un conjunto de funciones matemáticas, estadísticas y de texto que pueden ser usadas en un procedimiento.

En esta tabla el rango de los datos numéricos se denomina DATO. La celda C3 contiene un número aleatorio entre 0 y 1. A14 = "Ilmer", B14 = "Cóndor", A15 = "Ilmer Cóndor Espinoza", A16 = "XP203".

Datos		Usado en Excel	Resultado	En VBA
		=ALEATORIO()	0.462140992	Rnd()
Tiene por nombre Dato	13	=PROMEDIO(Dato)	22.5	
	30	=SUMA(Dato)	225	Sum(...)
	29	=MEDIANA(Dato)	23	Median(...)
	28	=MODA(Dato)	23	Mode(...)
	19	=MEDIA.GEOM(Dato)	21.38691325	GeomeAn(...)
	14	=ENTERO(C3*383)	177	Int(...)
	17	=MAX(Dato)	30	Max(...)
	10	=MIN(Dato)	11	Min(...)
	13	=RAIZ(C3)	0.680084142	Sqrt(...)
	29	=SUMAPRODUCTO(Dato,Dato)	5471	SumProduct(...)
Ilmer	Cóndor	=CONTAR(Dato)	10	Count(...)
Cóndor Espinoza, Ilmer		=CONTARA(Dato)	10	Counta(...)
XP203		=CONTAR.SI(Dato,">20")	7	CountIf(...)
		=SUMAR.SI(Dato,">20",Dato)	183	SumIf(...)
		=LOG10(C3)	-0.334874704	Log10(...)
		=LN(C3)	-0.771077501	Ln(...)
		=EXP(-C3)	0.629698315	Exp(...)
		=CONCATENAR(B14,", ",A14)	Cóndor, Ilmer	Concatenate(...)
		=IZQUIERDA(A14,3)	Ilm	Left(...)
		=DERECHA(A14,2)	er	Right(...)
		=EXTRAE(A16,3,3)	203	Mid(...)
		=VALOR(C24)	203	Value(...)
		=LARGO(A15)	22	Len(...)
		=ESPACIOS(A16)	21/07/00	Trim(...)
		=TEXTO(C25,"##.##")	$ 203.	Text(valorNum,"$ ###.##")
		=TEXTO(C25,"DD/MM/YY")	21/07/00	Text(valorNum,"DD/MM/YY")
Lógicas		And, Or		

Ejemplo 8

Escriba un código que al ingresar dos números, los imprima en forma ordenada (creciente)

```
Sub Ordena01()

Dim Num1, Num2 As Integer

Num1 = InputBox("Ingrese el primer número")

Num2 = InputBox("Ingrese el segundo número")

If Num1 < Num2 Then

   MsgBox ("Numero menor: " & Num1 & Chr(13) + Chr(10) & "Número mayor: " & Num2)

Else

   MsgBox ("Numero menor: " & Num2 & Chr(13) + Chr(10) & "Numero mayenor: " & Num1)

End If

End Sub
```

Ejemplo 9

Codificar un procedimiento que lea tres números e imprima en pantalla en forma creciente de sus valores. Como se puede apreciar, la lectura del código no es tan simple; requiere de una dosis de concentración. Más adelante usaremos arreglos para ordenar grandes series.

```
Sub ordena02()

Dim Num1, Num2, Num3 As Double

Num1 = InputBox("Ingresa el primer número")

Num2 = InputBox("Ingresa el segundo número")

Num3 = InputBox("Ingresa el tercer número")

If Num1 < Num2 Then

      If Num2 < Num3 Then
```

```
            MsgBox ("Los números ordenados: " + Chr(13) + Chr(10) & Num1 & Chr(13)
+ Chr(10) & Num2 & Chr(13) + Chr(10) & Num3)

         Else

            MsgBox ("Los números ordenados: " + Chr(13) + Chr(10) & Num1 & Chr(13)
+ Chr(10) & Num3 & Chr(13) + Chr(10) & Num2)

         End If

   Else

      If Num2 < Num3 Then

         If Num1 < Num3 Then

            MsgBox ("Los números ordenados: " + Chr(13) + Chr(10) & Num2 & Chr(13)
+ Chr(10) & Num1 & Chr(13) + Chr(10) & Num3)

         Else

            MsgBox ("Los números ordenados: " + Chr(13) + Chr(10) & Num2 & Chr(13)
+ Chr(10) & Num3 & Chr(13) + Chr(10) & Num1)

         End If

      Else

        MsgBox ("Los números ordenados: " + Chr(13) + Chr(10) & Num3 & Chr(13) +
Chr(10) & Num2 & Chr(13) + Chr(10) & Num1)

      End If

   End If
```

```
   End Sub
```

SENTENCIA CONDICIONAL SELECT CASE

Esta sentencia permite ejecutar una o más sentencias según el valor que tenga determinada variable. A diferencia de la sentencia IF, esta puede presentar muchas opciones del mismo nivel.

Sintaxis

```
SELECT CASE VarIndice
        Case Valor1:
            [ Sentencias ]
        Case Valor2, Valor3
            [ Sentencias ]
        ……
        Case Else
            [ Sentencias ]
End Select
```

Ejemplo 10

El siguiente procedimiento, luego de leer un valor desde el teclado, emite un mensaje.

```
Sub SelCase()
Dim Indice
Indice = InputBox("Ingrese un número entre 1 y 10")
Numero = 8
Select Case Indice
Case 1 To 4
   MsgBox ("Entre 1 y 4")
Case 5, 7, 9
   MsgBox ("Entre 5 y 9, pero no 6 ni 8")
Case 6
   MsgBox ("Es un número igual a 6")
Case Else
```

```
    MsgBox ("Es un número 8 ó  10")

End Select
```

```
End Sub
```

Ejemplo 11

El siguiente procedimiento genera dos números aleatorios, los multiplica por una constante y los emite en orden creciente.

```
Sub PMaxMin()

Dim Num1, Num2, Num3 As Double

Num1 = Rnd() * 8273

Num2 = Rnd() * 8273

MsgBox ("Los datos son: " & Num1 & " y " & Num2)

Select Case Num1

Case Is < Num2

      MsgBox (Num1 & " < " & Num2)

Case Else

      MsgBox (Num2 & " < " & Num1)

End Select
```

```
End Sub
```

Unidad 4. VBA. Más estructuras

Sentencias repetitivas

SENTENCIA FOR … NEXT

Permite ejecutar un determinado número de veces el grupo de sentencias incluidos en el cuerpo del For … Next

Sintaxis

FOR VarIndice = ValInit TO ValFInal [STEP Incr]

[Sentencias]

NEXT

Para ello requiere de una variable que funciona a modo de índice (VarIndice) el cual empieza en un primer valor (ValInit), por cada iteración que se realiza, se incrementa en una determinada cantidad (Incr), hasta llegar al extremo (ValFinal), después del cual, continua con la siguiente sentencia, debajo de Next. Si el incremento es la unidad, no se usa STEP Incr

Ejemplo 12

Se desea obtener la suma de los cuadrados de los 20 primeros números de 1 a 20.

Definiremos las siguientes variables:

I : Para la variable índice

Suma : Para contener la suma: Suma = Suma + I.^2

```
Sub Suma01()

Dim I  As Variant

Dim Suma As Double

Suma = 0

For I = 1 To 20

    Suma = Suma + I^2
```

```
Next

MsgBox("La suma de los primeros 20 números es: " & Suma)
```

```
End Sub
```

Ejemplo 13

Los siguientes datos corresponden a las superficies ocupadas por un conjunto de viviendas recién construidas. Se trata de determinar si el precio de la vivienda está en función a la superficie y en qué grado de correlación está.

Los datos son los siguientes:

Área (Y)	Valor (X)
100	40
140	49
150	54
150	56
130	48
120	46

	10	41
0		
	16	56
0		
	18	62
0		
	14	50
0		
	12	45
0		

Se trata de obtener las siguientes estadísticas: El promedio de superficie por vivienda y el valor promedio, el modelo lineal que exista.

Para ello tenemos las siguientes fórmulas: La ecuación a ser estimada es: $Y = \beta_0 + \beta_1 X$

$$\overline{X} = \frac{\sum X}{n} \qquad \overline{Y} = \frac{\sum Y}{n} \qquad \beta_1 = \frac{n\sum XY - \sum X \sum Y}{n\sum X^2 - (\sum X)^2} \quad \beta_0 = \overline{Y} - \beta_1 \overline{X}$$

A continuación expondremos el procedimiento

```
Sub Regre01()

Dim I, N As Integer

Dim SX, SX2, SY, SXY, Bo, B1 As Double

Dim MX, MY As Variant

' MX y MY contendrán la media de X e Y, respectivamente

' Se lee el número de datos a procesar

N = InputBox("Numero de datos:")
```

```
' Inicialización de las sumas
SX = 0: SY = 0: SX2 = 0: SXY = 0
For I = 1 To N
    Cadena = InputBox("Ingrese el para de datos, separados por coma")
    X = Val(Left(Cadena, 3))
    Y = Val(Right(Cadena, 2))
    SX = SX + X
    SY = SY + Y
    SX2 = SX2 + X ^ 2
    SXY = SXY + X * Y
Next
MX = SX / N
MY = SY / N
B1 = (N * SXY - SX * SY) / (N * SX2 - SX ^ 2)
Bo = MY - B1 * MX
' Emisión de resultados
MsgBox ("Total de datos: " & N)
MsgBox ("Superficie media = " & MX)
MsgBox ("Valor promedio = " & MY)
MsgBox ("Coeficiente Bo = " & Bo)
MsgBox ("Coeficiene de regresión = " & B1)
MsgBox ("La ecuacionde regresión es: Y = " & Bo & " + " & B1 & " X")
End Sub
```

SENTENCIA WHILE … WEND

Esta instrucción permite ejecutar un conjunto de sentencias incluidas en su ámbito, hasta que alguna condición se cumpla.

Sintaxis

WHILE Condicion

[Sentencias]

WEnd

Al interior del bucle del While debe haber alguna forma de actualizar la "Condicion" a fin de permitir que el While termine en algún momento.

Ejemplo 14

El siguiente ejemplo permite ingresar una serie de números hasta presionar <Enter>, en cuyo caso imprime la suma de todos ello.

```
Sub DoWhile()
Ix = Val(InputBox("Ingeresa un número; para terminar, presiona <Enter>"))
Suma = 0
While Ix > 0
  Suma = Suma + Ix
  Ix = Val(InputBox("Ingeresa un número; para terminar, presiona <Enter>"))
Wend
MsgBox ("La suma obtenida es = " & Suma)
End Sub
```

Ejemplo 15

Escriba un procedimiento que permita extraer la primera palabra dentro de un texto.

Solución

Leeremos carácter por carácter hasta encontrar un espacio en blanco, luego se imprimirá lo extraído

El código es el siguiente:

```
Sub dd()

Dim cadena As Variant

cadena = "Condor Espinoza, Ilmer"

x = Mid(cadena, 1, 1)

I = 1

xc = ""

While x <> " "

   xc = xc + x

   I = I + 1

   x = Mid(cadena, I, 1)

Wend

MsgBox xc

End Sub
```

Ejemplo 16

Escriba un procedimiento que al ejecutarse realice las operaciones básicas de una calculadora; es decir, sume (+), reste (-), multiplique (*), divida (/) y eleve a una potencia (^).

El esquema del trabajo es el siguiente:

Primero : Ingresar un número: **Op**

Segundo : Ingresar un código de operación: **Code**

Tercero : Mientras el valor de Code no sea "=", se pide otro número que ejecuta la operación indicada por Code, con el número previamente ingresado. Si se ha digitado "=", sale del bucle del While y emite el resultado.

El código es el siguiente:

```
Sub Calculator()

Dim Op As Double

'       Lee el primero valor

Op = Val(InputBox("Ingrese un número"))

'       Lee el código de operación

Code = InputBox("Codigo de operación")

'       Va a iterar mientras el valor de Code no sea "=".

While Code <> "="

   Select Case Code

      Case "+"

         Op = Op + Val(InputBox("Digite el número"))

      Case "-"

         Op = Op - Val(InputBox("Digite el número"))

      Case "*"

         Op = Op * Val(InputBox("Digite el número"))

      Case "/"

         Op = Op / Val(InputBox("Digite el número"))

      Case "^"

         Op = Op ^ Val(InputBox("Digite el número"))

      Case Else
```

```
            MsgBox ("Código inválido. Reinicie todo...")
            End
        End Select
        Code = InputBox("Código de operación")
    Wend
    MsgBox ("Resultado = " & Op)
End Sub
```

SENTENCIA DO … LOOP UNTIL

Esta instrucción permite ejecutar un grupo de sentencias contenidas en el ámbito, hasta que determinada condición se cumpla.

La condición que determinar el término de las iteraciones debe ser resultado de algún cálculo al interior del ámbito o debe haber sido leído en él.

Sintaxis:

```
DO
    [ Sentencias ]
LOOP UNTIL Expresión_de_Comparación
```

Como decíamos, Expresión_de_Comparación debe contener una relación lógica capaz de ser verdadera o falsa.

Observación:

Tome nota de la diferencia en el uso de la instrucción FOR, WHILE y DO. El uso de cada una de ellas es diferente y se emplea en diferentes contextos.

La instrucción FOR se emplea para repetir un bucle un número determinado de veces.

La instrucción WHILE se emplea para repetir el bucle siempre que la condición se cumpla.

La sentencia DO es lo mismo que WHILE, excepto que While pregunta antes de ejecutar el bucle, mientras que Do ejecuta el bucle y luego verifica la condición.

Ejemplo 17

Reescriba el procedimiento de la calculadora usando la instrucción DO … Loop Until

Como se puede apreciar, sólo cambiamos la instrucción While y la condición la hemos pasado al final, luego de leer el código de operación para una siguiente operación.

```
Sub Hasta()

Dim Op As Double

Op = Val(InputBox("Ingrese un número"))

Code = InputBox("Codigo de operación")

Do

   Select Case Code

      Case "+"

         Op = Op + Val(InputBox("Digite el número"))

      Case "-"

         Op = Op - Val(InputBox("Digite el número"))

      Case "*"

         Op = Op * Val(InputBox("Digite el número"))

      Case "/"

         Op = Op / Val(InputBox("Digite el número"))
```

```
        Case "^"
            Op = Op ^ Val(InputBox("Digite el número"))
        Case Else
            MsgBox ("Código inválido. Reinicie todo...")
            End
    End Select
    Code = InputBox("Código de operación")
Loop Until Code = "="
MsgBox ("Resultado = " & Op)
End Sub
```

Ejemplo 18

El siguiente código permite asignar a dos variables dos valores ingresados como una cadena de caracteres. Para ello, mediante el uso de While, almacena todos los caracteres en una variable. Luego otro While para extraer los caracteres del segundo valor.

El código es el siguiente:

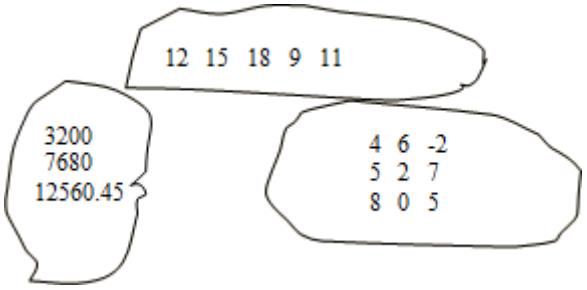
```
Sub ee()
cadena = "120, 3500.45"
Cad = ""
I = 1
Xc = Mid(cadena, 1, 1)
While Xc <> ","
    Cad = Cad + Xc
    I = I + 1
```

```
    Xc = Mid(cadena, I, 1)
  Wend
  Valor1 = Val(Cad)
  Cad = ""
  I = I + 1
  Xc = Mid(cadena, I, 1)
  While Xc <> "" Or I < Len(cadena)
    Cad = Cad + Xc
    I = I + 1
    Xc = Mid(cadena, I, 1)
  Wend
  Valor2 = Val(Cad)
  MsgBox Valor1 & "   " & Valor2
```

```
End Sub
```

Arreglos (Vectores o Matrices) en VBA

Un Arreglo es un conjunto de valores agrupados como fila / columna o como fila y columna.



Cuando el arreglo está agrupado por fila o columna se tiene un vector, desde el punto de vista matemático, mientras que está agrupado por un conjunto de filas y columna, se tiene una matriz.

El esquema anterior es una representación de dos tipos de objetos definidos como arreglos: lineales o cuadráticos.

Los arreglos pueden tener una o más dimensiones. Matemáticamente los arreglos unidimensionales constituyen los vectores y los multidimensionales, las matrices.

En términos de programación por computadora, es más común el uso de arreglos uni y dimensionales, en algunos casos, los tridimensionales.

En VBA los arreglos tienen un nombre pues se consideran un tipo de variable. Así como cada variable requiere de una localidad de memoria para almacenar su valor, así también, cada uno de los elementos del arreglo requieren de una localidad de memoria para almacenar su valor. Por esta razón, a los elementos de un arreglo se denotan de acuerdo al siguiente criterio:

M(1), M(2), … , M(10) corresponde a un arreglo lineal de 10 elementos

X(1,1)	X(1,2)	X(1,3)	X(1,4)	(X(1,5)
X(2,1)	X(2,2)	X(2,3)	X(2,4)	(X(2,5)

Es un arreglo bidimensional de 2 files con 5 columnas.

Todo tipo de arreglo usado en VBA debe ser declarado mediante la instrucción DIM

Ejemplo de declaración de arreglos

DIM X(10), Y(10) AS INTEGER

En este ejemplo se está declarando a X e Y como dos arreglos de tamaño 10 cada uno. Sus elementos se designarán por X(1), X(2), …, X(10), En el caso de Y, tendremos Y(1), Y(2), .., Y(10). En ambos casos se separan 10 localidades de memoria.

En la declaración

DIM Datos(80, 5), C(8, 5) , AS Integer

En este ejemplo el arreglo Datos tiene 80 elementos por fila y 5columas y el arreglo C contiene 8 filas y 5 columnas. En este caso, para Datos se separan 400 localidades de memoria y para C, 40 localidades.

Nota:

Si en el caso del arreglo X, se intenta usar el elemento X(11), se estará usando un subíndice fuera de límite. Esto significa que los arreglos deben ser declarados tomando en cuenta su máximo tamaño.

Ejemplo 19

Escriba Un programa que lea un conjunto de 10 datos numéricos, los almacene en un arreglo unidimensional y luego los imprima.

Solución

Sea Datos el nombre del arreglo. Supondremos que tiene 50 elementos. Usaremos la sentencia DIM para definir el arreglo como de tipo Integer.

Ya que se trata de introducir 10 datos, usaremos For … Next para ingresar los datos repetidamente. La variable de conteo en el For será I, no será necesario definirla.

Usaremos FOR ::: NEXT para leer los datos y otro para imprimir los valores leídos. Aunque bien podríamos leer e imprimirlos dentro del mismo FOR … NEXT.

Usaremos a InputBox(…) como función para ingresar los datos

El código es el siguiente:

```
Sub Arreglos01()

Dim Datos(10) As Integer

'   Lectura de datos

For I = 1 To 10

    Datos(I) = InputBox("Ingrese el dato: ")

Next

'   Impresión de los datos

MsgBox ("Estos son los datos leídos:")

For I = 1 To 10

    MsgBox (Datos(I))

Next
```

```
End Sub
```

Ejemplo 20

Escriba un procedimiento que permita obtener la suma y el promedio de un conjunto de n datos, leídos desde el teclado.

Solución

Ante todo, se debe ingresar los datos a ser procesados.

Como en el ejemplo anterior, usaremos *Datos* como nombre del arreglo que recibirá los datos ingresados; usaremos también la variable *Suma* que contendrá la suma de todos los datos ingresados y *Prom*, la variable que contenga el promedio.

Luego de leer los datos, usaremos otro FOR … NEXT para calcular la suma de los elementos; aunque se podría realizar la suma a la vez que se van leyendo los datos.

```
Sub Arreglos02()

Dim Datos(10) As Double

Dim Suma, Prom As Double

Suma = 0

N = InputBox("Numero de datos a ser leídos")

For I = 1 To N

   Datos(I) = InputBox("Ingrese el dato: ")

Next

'  Obtención de la suma

For I = 1 To N

   Suma = Suma + Datos(I)

Next

'          Cálculo del promedio

Prom = Suma/N
```

```
'          Impresión de resultados

MsgBox("La suma es: " & Suma & " ,   el promedio es: " & Prom)
```

```
End Sub
```

Ejemplo 21

Escriba un procedimiento que resuelva el problema planteado en el ejemplo de la página 31, usando arreglos. Ingrese los datos, separados por coma.

```
Sub Arreglo03()

Dim I, N As Integer

Dim SX, SX2, SY, SXY, Bo, B1 As Double

Dim MX, MY As Variant

'   Inicialización de las sumas

SX = 0: SY = 0: SX2 = 0: SXY = 0

'   Ingreso del número de datos

N = InputBox("Ingrese el número de datos a procesar")

For I = 1 To N

    Cadena = InputBox("Ingrese el para de datos")

    L = Len(Cadena)

'   Extrae el primer número y almacena en el vector MatY

    Y = ""

    Xc = Mid(Cadena, 1, 1)

    JPos = 1

    While Xc <> ","

        Y = Y + Xc

        JPos = JPos + 1
```

```
        Xc = Mid(Cadena, JPos, 1)
    Wend
    MatY(I) = Val(Y)
'   Extrae el segundo número y lo almacena en el vector MatX
    X = ""
    JPos = JPos + 1
    Xc = Mid(Cadena, JPos, 1)
    While JPos <= L
        X = X + Xc
        JPos = JPos + 1
        Xc = Mid(Cadena, JPos, 1)
    Wend
    MatX(I) = Val(X)
Next
'   Cálculos
For I = 1 To N
    SX = SX + MatX(I)
    SY = SY + MatY(I)
    SXY = SXY + MatX(I) * MatY(I)
    SX2 = SX2 + MatX(I) * MatX(I)
Next
'   Estimacion de los coeficientes
B1 = (N * SXY - SX * SY) / (N * SX2 - SX * SX)
```

```
Bo = SY / N - B1 * SX / N

MsgBox ("La ecuación de regresión estimada es:")

MsgBox ("Y = " & Bo & " + " & B1 & " X")
```

```
End Sub
```

Observación:

En el ejemplo anterior, como verá, hemos extraído los caracteres contenido en la variable "cadena", carácter por carácter hasta encontrar ",". Repite el mismo procedimiento para extraer el segundo dato.

Más adelante veremos otra forma de extraer los números (Ver Ejemplo 8 de la sección Objeto Range).

Procedimientos que transfieren el control a otros procedimientos

Desde un procedimiento podemos efectuar llamadas a otros procedimientos o podemos transferir valores hacia otros procedimientos.

Ejemplo 22

El siguiente procedimiento PrPrin, llama al procedimiento Lectura que se encarga de leer el nombre de los meses (separado por coma) en una cadena, llama al procedimiento Decode que se encarga de almacenar los nombres de los meses en un arreglo unidimensional y termina llamando al procedimiento Prin que se encarga de imprimir el arreglo de los nombres de los meses.

Nota:

Para que los datos sean compartidos por todos los procedimientos, usamos DIM, externo a todo procedimiento, en forma global.

```
Dim Meses(12) As String

Dim Cadena As Variant
```

```
Sub PrPrin()

'   Llama al procedimiento Lectura

Lectura
```

```
    '   Llama al procedimiento Decode

    Decode

    '   Llama al procedimiento Prin

    Prin

    End Sub

    Sub Lectura()

    Cadena = InputBox("Ingrese los nombres de los meses, separado por coma", "Decodificación de meses")
```

```
    End Sub
```

```
    Sub Decode()

    L = Len(Cadena)

    I = 0

    IPos = 1

    Xmes = ""

    While IPos <= L

        If Mid(Cadena, IPOs, 1) = "," Or IPos = L Then

            I = I + 1

            Meses(I) = Xmes

            Xmes = ""

        Else

            Xmes = Xmes + Mid(Cadena, IPos, 1)

        End If

        IPos = IPos + 1
```

```
Wend

End Sub

Sub Prin()

For I = 1 To 12

   MsgBox ("Mes: " + Meses(I))

Next

End Sub
```

Llamada y transferencia de datos y resultados entre procedimiento

El siguiente ejemplo ilustra la forma cómo se llama a un procedimiento que se encarga de leer dos datos, se llama otro procedimiento que calcula el producto de los valores leídos, se imprime en el llamador los valores leídos y se llama a otro para que imprima el resultado devuelto por el procedimiento de cálculo.

Ejemplo 23

```
Sub Transf()

Call Entrada(a, b)

Call Procesa(a, b, c)

MsgBox ("En el procedimiento principal: " & a & " , " & b)

Call Imprime(c)

End Sub

Sub Entrada(x, y)

x = InputBox("X = ")

y = InputBox("Y = ")

End Sub

Sub Procesa(x, y, r)
```

```
    r = x * y

    End Sub

    Sub Imprime(z)

    MsgBox ("Calculado en un procedimiento e impreso en otro: " & z)

    End Sub
```

Funciones

Como hemos dicho antes, un procedimiento también puede estar constituido por una función.

La sintaxis de una función es:

```
Function NombreDeFuncion(Arg1, Arg2, …, Argk)

        [ Sentencias ]

End Function
```

La diferencia entre el procedimiento estándar que hemos descrito y este otro que es una función está en que un procedimiento función devuelve un valor (aunque sea nulo), mientras que un procedimiento estándar no devuelve valor alguno.

El hecho que un procedimiento pase algunos valores a otro no es lo mismo que la capacidad que tienen las funciones de devolver un resultado.

Tanto un procedimiento que devuelve un resultado como una función que devuelve un resultado tienen su semejanza y su diferencia de uso, dependiendo del contexto en donde se encuentran.

Ejemplo 24

El siguiente procedimiento *ProcPrin* llama a otro procedimiento *Lee*, que se encarga de leer dos valores, e imprime el valor devuelto por la función *Decide* quien se encarga de determinar si se debe sumar o multiplicar los valores leídos.

El código es el siguiente:

```
Sub ProcPrin()
```

```
    Dim a, b As Double

    Call Lee(a, b)

    MsgBox (Decide(a, b))
```

```
End Sub

Sub Lee(x, y)

x = InputBox("Primer valor: ")

y = InputBox("Segundo valor: ")
```

```
End Sub

Function Decide(r, s As Double)

If r < s Then

   Decide = r + s

Else

   Decide = r * s

End If
```

```
End Function
```

Creación de funciones de usuario ejecutados desde Excel.

Como bien sabemos el Excel tiene muchas funciones organizadas por categorías que nos permiten resolver una diversidad de problemas. Estas funciones van desde aquellas que resuelven problemas matemáticos hasta problemas financieros pasando por aquellos problemas de manejo de texto, estadísticos, de base de datos, decisiones y fechas.

Y si estas funciones no resuelven todos nuestros problemas, podemos construir nuestras propias funciones y añadirlas al conjunto de las que posee el Excel. La categoría a las que pertenecen las funciones que, como usuario podamos crear, constituye las **funciones definidas por el usuario**.

Una función definida por el usuario no es otra cosa que un procedimiento especial denominado función, pues devuelve un resultado.

Una vez codificada esta función, el uso que se hace de ellas es exactamente igual al de las otras funciones; es decir, al digitar su nombre en una celda con sus respectivos parámetros, estaremos en capacidad de recibir el resultado.

Ejemplo 25

Escriba una función que devuelva el área de un rectángulo dado su base y altura.

Solución:

Nombre de la función : Area

Argumentos : Base, Altura

La función calculará: Base*Altura/2

El código es el siguiente:

Function **Area**(Base,Altura)

Area = Base*Altura

End Function

Ahora en Excel,

• Use el Asistente para funciones

• Seleccione la categoría "Definidas por el usuario"

• Seleccione la función cuyo nombre sea Area

• Haga clic en <Aceptar>

• En cada parámetro ingrese las celdas que contienen el valor correspondiente: En Base: B9; en Altura: B10.

• Haga clic en <Aceptar>

Estando en Excel, puede digitar =Area(B9,B10) en la celda B11 y presionar <Enter>

Ejemplo 26

Escriba una función que permita obtener el perímetro o área de un trapecio teniendo como datos las dos bases y la altura. Si el trapecio no es isósceles debe devolver el mensaje "No hay datos para obtener el perímetro de un trapecio que no es isósceles".

Solución

```
Function Trapecio(Bmay, Bmen, H, Ians)

Select Case Ians

   Case "p", "P"

      If (Bmay = 2 * Bmen) Then

         Trapecio = Bmay + Bmen + 2 * Sqr(H * H + Bmen ^ 2 / 4)

      Else

         MsgBox ("No se tiene datos para calcular el área" + Chr(13) + Chr(10) + "de un trapecio que no es isósceles")

      End If

   Case "a", "A"

      Trapecio = H / 2 * (Bmay + Bmen)

End Select

End Function
```

Ejemplo 26

Dada la ecuación cuadrática $a x^2 + bx + c = 0$, las raíces de esta ecuación son:

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \qquad \text{y} \qquad x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

Estas raíces existen siempre que el valor del radical es no negativo.

Escriba una función que permita calcular las raíces de una ecuación cuyos coeficientes son conocidos y se ingresan en tres celdas de una hoja.

Solución

Daremos por nombre a la función: FRaiz1 aquella función que nos devuelve la primera raíz y FRAiz2 a la que devuelve la segunda raíz. En cuanto a los parámetros serán A, B y C. El código correspondiente es el que sigue.

```
Function FRaiz1(a, b, c)
If (b * b - 4 * a * c < 0) Then
    MsgBox "Las raíces son imagináreas"
    Return
Else
    FRaiz1 = (-b + Sqr(b * b - 4 * a * c)) / (2 * a)
End If
End Function
```

```
Function FRaiz2(a, b, c)
If (b * b - 4 * a * c < 0) Then
    MsgBox "Las raíces son imagináreas"
    Return
Else
    FRaiz2 = (-b - Sqr(b * b - 4 * a * c)) / (2 * a)
End If
End Function
```

Cómo usar esta función de usuario

Una vez que ha creado una determinada función, para verificar su adición al conjunto de funciones, haga uso del asistente y seleccione la categoría de "Funciones definidas por el usuario". Allí encontrará una lista de ellas.

Para usarla, puede hacer uso del asistente de funciones o digitar en la celda donde desee el resultado como sigue:

```
= FRaiz1(A1,B1,C1)          ' Suponiendo que A, B y C están en A1, B1 y C1

= FRaiz2(A1,B1,C1)          ' Suponiendo que A, B y C están en A1, B1 y C1
```

Con ello obtendrá la primera y segunda raíz de la ecuación respectiva.

Nota:

He aquí una función particularmente útil:

Ejemplo 27

Implemente una función que permita calcular la raíz n – ésima de un número. La función debe ser capaz de obtener raíces impares de valores negativos y emitir un mensaje si se pretende obtener raíz par de un valor negativo.

Solución

Llamaremos r a la función que recibirá dos parámetros: **A** que contendrá el número a quien se le saque la raíz y **N** que será la cantidad radical.

Primero debe verificar si el número a ser usado como cantidad radical par o no. Si es par debe verificar si el valor es negativo. Si así fuera, devuelve la raíz pedida, en caso contrario, emite un mensaje de error pues no se puede obtener un radical par de un valor negativo.

La función es la siguiente:

```
Function r(a, n)

If n = Int(n / 2) * 2 Then

   If a >= 0 Then

      r = a ^ (1 / n)

   Else

      MsgBox "No se puede obtener raíz real par, de un valor negativo"

   End If
```

```
    Else
        If a < 0 Then
            r = -Abs(a) ^ (1 / n)
        Else
            r = a ^ (1 / n)
        End If
    End If
```

```
End Function
```

Ahora codificaremos una función que sea usada por un procedimiento.

Ejemplo 28

Escriba un procedimiento para resolver problemas de conteo o enumeración. El procedimiento debe tener una rutina principal donde se decida qué tipo de técnica se usará.

Solución

El procedimiento principal se llamará : MainTc

La función que calcula el factorial se llamará : Factorial

La función que calcule las permutaciones se llamará : Permut

La función que calcule las permutaciones con repetición: : PermutRep

La función que calcule las combinaciones se llamará : Combinat

La rutina principal se encargará de pedir que se ingrese el código de operación a realizar. 1 = Factorial; 2 = Permutaciones; 3 = Permutaciones con repetición; 4 = Combinaciones

A continuación pedirá que se ingrese el o los valores según la operación seleccionada.

La sentencia Select Case se encarga de decodificar la operación deseada.

Observe cómo se pide el ingreso de los datos, dentro de los argumentos que se pasan a la función.

A continuación se da el código:

```
Sub MainTc()

Code = InputBox("Digite el numeral deseado: " + Chr(13) + Chr(10) _

+ "1. Factorial F(n)" + Chr(13) + Chr(10) + "2. Permutaciones P(n,m)" + Chr(13) + Chr(10) _

+ "3. Permutaciones con repetición Pr(n,m)" + Chr(13) + Chr(10) + "4. Combinaciones C(n,m)")

Select Case Code

    Case 1:

        Result = Factorial(InputBox("Factorial de: "))

    Case 2:

    Result = Permut(InputBox("Permutaciones de n = "), InputBox("tomados de m = "))

    Case 3:

    Result = PermutRep(InputBox("Permut. con repet. de n = "), InputBox("tomados de m = "))

    Case 4:

        Result = Combinat(InputBox("Combinaciones de n = "), InputBox("tomados de m = "))

End Select

MsgBox Result
```

```
End Sub

Function Factorial(n)

Factorial = n
```

```
If Factorial = 1 Then
Else
    Factorial = Factorial * Factorial(n - 1)
End If
```

```
End Function
```

```
Function Permut(n, m)
If n >= m Then
    Permut = Factorial(n) / Factorial(n - m)
Else
    MsgBox "Error en datos..."
End If
```

```
End Function
```

```
Function PermutRep(n, m)
PermutRep = n ^ m
```

```
End Function
```

```
Function Combinat(n, m)
If n >= m Then
    Combinat = Factorial(n) / (Factorial(m) * Factorial(n - m))
Else
    MsgBox "Error en datos..."
End If
```

```
End Function
```

Funciones para manejo de texto

=Left(Texto,NChar)

Esta función permite extraer los primeros "NChar" caracteres de la cadena "Texto"

=Right(Texto, NChar)

Contrario a la anterior, esta función permite extraer de la cadena "Texto", los úiltimos "NChar" caracteres.

=Mid(Texto, Init, NChar)

Esta función permite extraer de Texto, "NChar" caracteres a partir del carácter "Init"

=Instr(Texto1, Texto2,[Inicial])

Esta función devuelve la posición inicial en que se encuentra la cadena Texto2, dentro de la cadena Texto1. Esto lo hace examinando a partir de la posición Inicial.

Ejemplo

```
Sub Apellidos_Y_Nombres()


Texto = InputBox("Ingresa tus apellidos y nombres (Ap Am, Nombres)")

ApPaterno = Left(Texto, InStr(Texto, " "))

ApMaterno = Mid(Texto, InStr(Texto, " "), Len(Texto) - InStr(Texto, ",") - 6)

Nombres = Right(Texto, Len(Texto) - InStr(Texto, ","))


MsgBox ApPaterno

MsgBox ApMaterno

MsgBox Nombres


End Sub
```

Unidad 5. Programación usando objetos del Excel

Objetos del Excel

Advertencia:

Los siguientes esquemas gráficos han sido extraídos de la ayuda del Editor de Visual Basic para Aplicaciones. Creemos que es la manera más didáctica de presentar los diferentes objetos del Excel y su ubicación dentro de la estructura de dicha estructura.

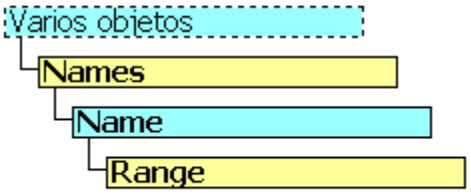
El programa Excel dispone de una gran cantidad de objetos. Muchos de estos objetos forman parte de una clase o colección. La tabla presentada en la siguiente página, muestra la estructura de los objetos y colecciones en Excel.

En esta tabla se observa que las colecciones están con fondo gris con poca tonalidad y los objetos en un color gris con más tonalidad.

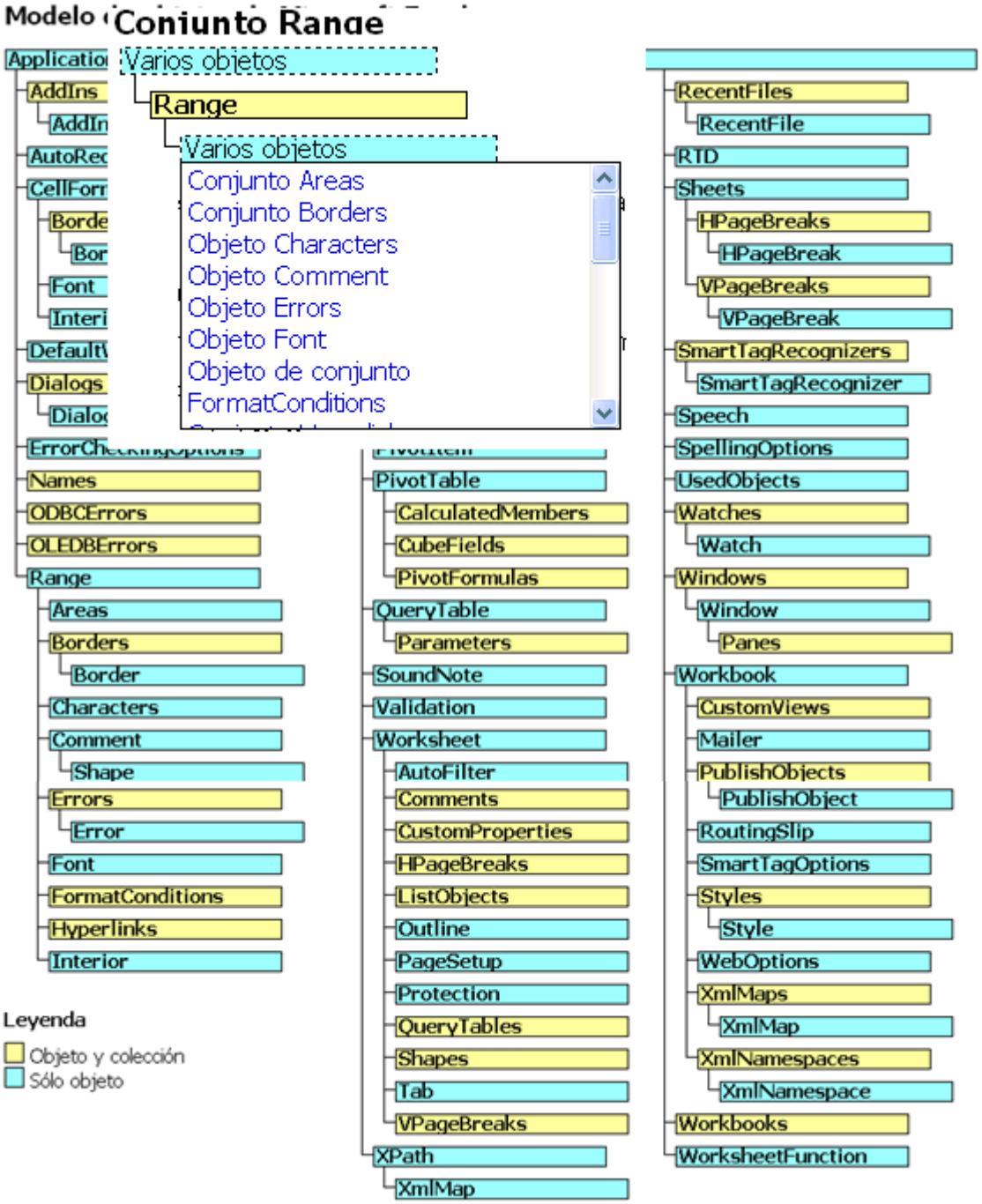
Se puede apreciar que dentro del objeto **CellFormat** se tiene la colección o conjunto **Borders** y dentro de éste, el objeto **Borde**.

En el caso de la colección o conjunto Names, tenemos la siguiente subestructura:

Objeto de conjunto Names



Por otro lado, el conjunto Range contiene un conjunto de objetos que se muestra en el siguiente esquema.



De la observación de esta tabla, podemos decir que el objeto **Application** proporciona un contenedor de toda la aplicación y cada objeto WorkBook contiene una cantidad de objetos **WorkSheet**; dentro de este objeto referirnos al objeto **Range**, el cual nos permitirá trabajar con celdas o grupos de celdas.

Las clases más utilizadas en Excel, son las siguientes:

Objeto Application

Objeto WorkBook

Objeto WorkSheet

Objeto Range

Como se dijo antes, para usar algún método de acceso a un objeto o colección, haremos uso del punto, el cual vinculará los objetos y su colección y el método mediante el cual accederemos a las propiedades de los objetos.

En este caso, para acceder al objeto Font, usaremos la propiedad Font del conjunto Range, de la hoja 1, del libro Simple.xls

Usaremos la siguiente sintaxis:

Range("B5").Font.Bold = True

Esto permite poner la celda B5 en resaltado (negrita o Bold).

Expresiones equivalentes a esta, puede ser:

WorkSheets(1).Range("B5")).Font.Bold = True

O también

Application.WorkBooks("Simple.xls").WorkSheets(1).Range("B5").Font.Bold = True

Nota:

Nosotros usaremos la notación más sencilla, pero claro está, que para ello deberemos estar usando los conjuntos previos. Esto significa que, en el caso de la expresión

Range("B5").Font.Bold = True

Debe estar en uso, el libro **Simple.xls**, debemos haber seleccionado y activado previamente la hoja 1.

A continuación estudiaremos la forma de acceder y/o usar estas colecciones u objetos mediante el lenguaje VBA. Según corresponda, modificaremos el programa de las macros a fin de dotarle de mayor potencia interactiva.

Objeto Application. Propiedades. Métodos. Ejemplos.

El objeto Application representa la propia aplicación Microsoft Excel. El uso de este objeto proporciona toda la información referida a la aplicación que está en uso. En muchos casos será necesario referirnos explícitamente a la aplicación en uso; en otras se puede prescindir de él.

El objeto Application contiene:

Valores y opciones de toda la aplicación. Por ejemplo, muchas de las opciones del cuadro de diálogo Opciones, del comando Herramientas.

Métodos que devuelven objetos de nivel superior, como ActiveRange, ActiveWorkBook, ActiveCell, ActiveChart, ActiveSheet, etc.

Sintaxis:

Calficador[.Valor u opción] [.Método]

Donde Calificador es Application.

Ejemplo 1

Application.WorkBooks.Close

En este ejemplo, se aplica la propiedad WorkBooks al ojeto Application. La propiedad WorkBooks es modificado por le ejecución del método Close. En consecuencia, esta sentencia permite cerrar el libro que está en uso.

Ejemplo 2

Application.Windows("Ventas.xls").Activate

Este ejemplo se aplica la propiedad Windows al objeto Application que está en ejecución.

Ejemplo 3

Set Xls = CreateObject("Excel.Sheet")

Xls.Application.WorkBooks.Open "Ventas.xls"

En este ejemplo se crea un objeto libro en otra aplicación y luego abre el libro existente, Ventas.xls.

Propiedades del objeto application

Las propiedades del objeto application se divide en cuatro grupos:

• Propiedades que controlan el estado

• Propiedades que controlan la presentación

• Propiedades que devuelven objetos

• Propiedades que controlan la manipulación de los archivos

Las propiedades que controlan el estado definen el ambiente donde se ejecuta el Excel.

La siguiente lista muestra algunas de las propiedades del objeto application

Propiedad	Tipo	Descripción
Cursor	XlMousePointer (xlDefault, xlIBeam, xlNorthwestArrow, xlWait)	Obtiene o establece el aspecto del puntero del *mouse* (ratón).
EditDirectlyInCell	Boolean	Obtiene o establece la capacidad de modificar celdas directamente en el lugar correspondiente. Si es False, las celdas sólo se pueden modificar en la barra de fórmulas.
FixedDecimal	Boolean	Si es True, todos los valores numéricos utilizan la propiedad **FixedDecimalPlaces** para determinar el número de decimales; en caso contrario, la propiedad **FixedDecimalPlaces** se omite (el valor predeterminado es False).
FixedDecimalPlaces	Long	Determina el número de decimales que se utilizan para los datos numéricos si la propiedad **FixedDecimal** es True.
Interactive	Boolean	Obtiene o establece la capacidad del usuario de interactuar con Excel a través del teclado o el *mouse*. Si establece esta propiedad en False, asegúrese de cambiarla de nuevo a True en el controlador de excepciones.
MoveAfterReturn	Boolean	Si es True, la selección se mueve a la siguiente celda al presionar ENTRAR; el valor predeterminado es True.
MoveAfterReturnDirection	xlDirection (xlDown, xlToLeft, xlToRight, xlUp)	Indica la dirección de movimiento después de presionar ENTRAR, si la propiedad **MoveAfterReturn** es True. El valor predeterminado es **xlDown**.
ScreenUpdating	Boolean	Si es True, Excel actualiza su pantalla después de cada llamada a un método. Puede desactivar la presentación mientras se ejecuta el código. Asegúrese de establecer de nuevo esta propiedad en True.
SheetsInNewWorkbook	Long	Obtiene o establece el número de hojas que Excel incluye automáticamente en los nuevos libros.
StandardFont	String	Obtiene o establece el nombre de la fuente predeterminada; no surte efecto hasta que se reinicia Excel.
StandardFontSize	Long	Obtiene o establece el tamaño de la fuente predeterminada de Excel; no surte efecto hasta que se reinicia Excel.
StartupPath (sólo lectura)	String	Devuelve la ruta de acceso completa de la carpeta que contiene los complementos de inicio de Excel.

TemplatesPath (sólo lectura)	String	Devuelve la ruta de acceso completa de la carpeta que contiene las plantillas.

La siguiente lista muestra las propiedades que controlan la presentación

Propiedad	Tipo	Descripción
DisplayAlerts	Boolean	Si es True (el valor predeterminado), Excel muestra mensajes de advertencia mientras se ejecuta el código, cuando sea necesario. Establezca esta propiedad en False para omitir las advertencias y seleccionar el valor predeterminado.
DisplayFormulaBar	Boolean	Si es True (el valor predeterminado), Excel muestra la barra de fórmulas estándar para la modificación de celdas; establézcala en False para ocultar la barra.
DisplayFullScreen	Boolean	Si es True, Excel se ejecuta en el modo de pantalla completa (con un efecto diferente al que se obtiene maximizando la ventana de Excel); el valor predeterminado es False.

En cuanto a las propiedades que devuelven objetos, se muestra en la siguiente lista:

Propiedad	Tipo	Descripción
ActiveCell	Range	Devuelve una referencia a la celda actualmente activa en la ventana activa (la ventana
ActiveChart	Chart	Devuelve una referencia al gráfico actualmente activo. Un gráfico incrustado sólo se considera activo cuando está seleccionado o activado.
ActiveSheet	Object	Devuelve una referencia a la hoja activa del libro activo.
ActiveWindow	Window	Devuelve una referencia a la ventana activa (la ventana que está delante); devuelve Nothing si no hay ninguna ventana activa.
Charts	Sheets	Devuelve una colección de objetos **Sheet** (el objeto principal de **Chart** y **Worksheet**) que contiene referencias a cada uno de los gráficos del libro activo.
Selection	Object	Devuelve el objeto seleccionado en la aplicación, como **Range**, **Worksheet** u otro objeto. Se aplica también a la clase **Window**, en donde la selección es normalmente un objeto **Range**. Si no hay ningún objeto actualmente seleccionado, devuelve Nothing.
Sheets	Sheets	Devuelve una colección de objetos **Sheet** con referencias a cada una de las hojas del libro activo.
Workbooks	Workbooks	Devuelve una colección de objetos **Workbook** con referencias a todos los libros abiertos.

Estas propiedades tienen mayor uso en una aplicación concreta. En general, todas estas propiedades hacen referencia a los objetos que devuelven objetos.

Por ejemplo en el caso de la propiedad Sheets, ésta devuelve un conjunto de objetos contenidos en la colección Sheets.

Usaremos estas propiedades cuando hagamos ejemplos relacionado con libros y hojas; es decir, con objetos WorkBook y objetos Sheets.

Métodos del objeto application

El objeto application dispone de algunos métodos que permiten realizar operaciones relacionadas con la aplicación activa.

Entre estos métodos tenemos:

Calculate

CheckSpelling

Evaluate

MailSystem

Quit

Undo

Cómo usarlos alguno de ellos:

ThisApplication.Calculate

Permite ejecutar todos los libros abiertos

ThisApplication.Quit

Permite salir del Excel mediante la programación

ThisApplication.Undo

Permite cancelar la última acción realizada por el usuario.

Objeto WorkBook. Propiedades y Métodos. Ejemplos.

El objeto WorkBook representa un libro en Excel. Como objeto, es un elemento de la clase o conjunto WorkBook. Como clase o conjunto, WorkBook contiene todos los objetos WorkBooks abiertos en un momento determinado en Microsoft Excel.

Muchos miembros de la clase Application se presentan también como miembros de la clase WorkBook.

En la clase WorkBook las propiedades se aplican a un libro específico en lugar de aplicarse al libro activo.

Propiedades del objeto WorkBook

Algunas de las propiedades del objeto WorkBook son las siguientes:

a) Propiedad WorkBooks

Devuelve todos los libros abiertos.

Se puede usar como

Application.WorkBooks.

O también simplemente

WorkBooks.

Ejemplo 01

Abrir el libro **Tempo.xls** que se encuentra en la unidad y carpeta en uso.

```
WorkBooks.Open FileName:="Tempo.xls"
```

Ejemplo 02

Abrir un libro que se encuentra en una unidad y carpeta que no está en uso.

```
Workbooks.Open Filename:="G:/Libros/Progmacros/Pedidos.xls"
```

Abre el archivo **Pedidos.xls** que se encuentra en la carpeta ProgMacros dentro de la carpeta Libros, de la unidad G. Como se puede apreciar, el nombre del libro viene precedida por la ruta donde se encuentra dicho libro-

Ejemplo 03

El siguiente ejemplo permite activar el segundo libro que ya está abierto.

```
WorkBooks(2).Activate
```

Ejemplo 04

El siguiente ejemplo abre el libro "Libro1.xls", imprime en pantalla el nombre del libro abierto, abre también el libro "libro2.xls"; activa el primero y cierra el segundo. Para

ejecutarlo, debe estar abierto un libro cualquiera diferente a Libro1 y Libro2, conteniendo la macro.

```
Sub Libros01()

Workbooks.Open ("C:/Libro1.xls")

MsgBox Workbooks("Libro1.xls").Name

Workbooks.Open ("C:/libro2")

Workbooks("Libro1.xls").Activate

Workbooks("Libro2.xls").Close

End Sub
```

b) Propiedad ActiveWorkBook

Esta propiedad devuelve el libro activo; es decir, hace referencia al libro activo de tal forma que cualquier acción que se pueda ejecutar, afectará al libro abierto y activo, dentro del grupo de libros que pudieran estar abiertos.

Ejemplo

El siguiente ejemplo imprime en pantalla el nombre del libro activo

```
MsgBox ("El nombre del libro activo es " & ActiveWorkBook.Name
```

Ejemplo

En el siguiente esquema se ejecuta una serie de acciones estando en uso el libro activo

```
With ActiveWorkBook
        Nombre     = .Name
        Ruta       = .Path
        HojaActiva = .Sheets(1).Select
        Proteger   = .Protect
End With
```

c) Propiedad Open

WorkBooks.Open Nombre

Permite abrir un libro existente (ya grabado) cuyo nombre es "Nombre".

Si el libro no se encuentra en la unidad y carpeta en uso, el nombre del libro debe contener la ruta donde se ubica el libro

Ejemplo

```
Sub OpenUp()
    Workbooks.Open("C:\MyFolder\MyBook.xls")
End Sub
```

Ejemplo 01

El siguiente ejemplo abre los libros Libro1.xls y Libro2.xls. La última instrucción permite crear otro nuevo libro que va a ser una copia exacta del libro que está activo. Según la secuencia, el libro activo es Libro2.xls; por lo que seguramente crea otro nuevo libro con el nombre definido por omisión.

```
Sub Libros02()
Workbooks.Open ("C:/Libro1.xls")
MsgBox Workbooks("libro1.xls").Name
Workbooks.Open ("C:/libro2")
ActiveWorkbook.Sheets.Copy
End Sub
```

Observe que en la penúltima instrucción hemos añadido un comentario al final de la misma mediante el uso del apóstrofe: **'**

Las siguientes instrucciones hace uso del libro activo

```
MagBox ActiveWorkbook.Name       Imprime el nombre del libro activo
ActiveWorkbook.SaveAs            Graba el libro activo
```

ActiveWorkbook.Close Cierra el libro activo

MsgBox ActiveWorkbook.Path Imprime la ruta en donde se encuentra el libro activo.

Ejemplo 02

El siguiente ejemplo permite duplicar un libro determinado

```
Sub Duplica()

Workbooks.Open ("C:/Libro1.xls")

ActiveWorkbook.Sheets.Copy

ActiveWorkbook.SaveAs "Kiko.xls"

ActiveWorkBooks.Close

WorkBooks("Libro1.xls").Close

End Sub
```

d) Propiedad Add

```
WorkBooks.Add
```

Esta propiedad permite crear un nuevo objeto libro el cual se convierte automáticamente en libro activo.

El siguiente ejemplo crea un nuevo libro y modifica alguna de las propiedades del nuevo libro

Ejemplo

```
Sub AddNew()

Set LibroNuevo = Workbooks.Add

    With LibroNuevo

        .Title = "Proforma 01"

        .Subject = "Consultas"

        .SaveAs Filename:="Proformas.xls"
```

```
    End With
End Sub
```

Objeto WorkSheet. Propiedades y Métodos. Ejemplos.

El objeto WorkSheet representa un libro en Excel. Como objeto, es un elemento de la clase o conjunto WorkSheets. Como clase o conjunto, WorkSheets contiene todos los objetos WorkSheet abiertos en un momento determinado en Microsoft Excel.

Cada objeto WorkSheet representa una hoja de cálculo.

Propiedades del objeto WorkSheet

Describiremos dos propiedades del objeto WorkSheet

a) Propiedad WorkSheets

Esta propiedad devuelve el resultado de una acción realizada sobre una determinada hoja de cálculo identificada por su nombre o por un índice en el argumento de esta propiedad.

Sintaxis:

WorkSheets(x).Método

Donde

x	puede ser el número de hoja o el nombre de la misma.
Método	es el método que se usará sobre la hoja x

b) Propiedad ActiveSheet

Esta propiedad permite hacer referencia a la hoja activa y extraer o modificar alguna de sus propiedades mediante el uso de algunos métodos.

Sintaxis:

ActiveSheet.Método

En los siguientes ejemplos haremos uso de estas dos propiedades sea en forma individual o en forma complementaria

Ejemplo 01

WorkSheets(1).Activate

Permite activar la hoja 1 del libro activo

Ejemplo 02

El siguiente ejemplo pide la ruta y nombre de libro a abrirse y activa la hoja 3 de dicho libro.

```
Sub Activar()

Libro = Trim(InputBox("Ruta y nombre del archivo"))

Libro = Libro + ".xls"

Workbooks.Open Filename:=Libro

Book = ActiveWorkbook.Name

Workbooks(Book).Worksheets("Hoja3").Activate

End Sub
```

Ejemplo 03

El siguiente ejemplo permite obtener el nombre de la hoja activa.

```
Sub Nombre()

Worksheets(1).Activate

NombreHoja = Worksheets(1).Name

MsgBox NombreHoja

End Sub
```

Ejemplo 04

En el siguiente ejemplo se activa la hoja 3 y se añade una nueva hoja a la izquierda de la hoja activa.

```
Sub NuevaHoja()

Worksheets(3).Activate
```

```
Worksheets.Add

End Sub
```

Ejemplo 05

El siguiente ejemplo activa la tercera hoja, añade una nueva hoja y le cambia el nombre por "Ventas" y luego imprime el nombre devuelto por el método **Name**

```
Sub NuevoNombre()

Worksheets(3).Activate

Worksheets.Add

NomHoja = ActiveSheet.Name

Worksheets(NomHoja).Name = "Ventas"

MsgBox ActiveSheet.Name

End Sub
```

Ejemplo 06

El siguiente ejemplo permite añadir una nueva hoja. Moverlo después de la hoja 1 e imprimir el número de hojas.

```
Sub MoverHoja()

Worksheets(3).Activate

Worksheets.Add

NomHoja = ActiveSheet.Name

Worksheets(NomHoja).Name = "Ventas"

Worksheets.Move After:=Worksheets(1)

MsgBox Worksheets.Count

End Sub
```

Ejemplo 07

En el siguiente ejemplo se oculta la hoja3 y luego de activar la hoja1, se cambia la orientación de la hoja y se imprime después.

```
Sub ImprimirHoja()

Worksheets(1).Visble = False

Worksheets("Hoja3").Activate

ActiveSheet.PageSetup.Orientation = xlLandscape

ActiveSheet.PrintOut

End Sub
```

Ejemplo 08

El Siguiente ejemplo añade una nueva hoja y luego muestra una lista de los nombres de las hojas, en el orden en el que se encuentran, contenidas en el libro activo.

```
Sub ListaHojas()

Dim Cadena As Variant

Cadena = ""

Set NuevaHoja = Sheets.Add(Type:=xlWorksheet)

For i = 1 To Sheets.Count

   Cadena = Cadena + Sheets(i).Name + Chr(10) + Chr(13)

Next i

   MsgBox Cadena

End Sub
```

Comentarios:

Primero hemos definido a la variable cadena como de tipo variant

La instrucción Set permite definir a NuevaHoja como un objeto WorkSheet.

Ejemplo 09

El siguiente ejemplo, además de hacer lo mismo que el ejemplo 8, activa la hoja 2 (que es la segunda en secuencia), deposita un texto en B2 y activa el objeto NuevaHoja que es una hoja.

```
Sub Lista()

Dim Cadena As Variant

Cadena = ""

Set NuevaHoja = Sheets.Add(Type:=xlWorksheet)

For i = 1 To Sheets.Count

   Cadena = Cadena + Sheets(i).Name + Chr(10) + Chr(13)

Next i

   MsgBox Cadena

Sheets(2).Activate

Range("B2") = "Hola estamos cenando"

NuevaHoja.Activate

Range("B3").Select

End Sub
```

Ejemplos sueltos:

La siguiente declaración mueve todas las hojas a la derecha de la última hoja.

```
Worksheets.Move After:=Sheets(Sheets.Count)
```

La siguiente declaración permite inserter tres nuevas hojas antes de la primera hoja.

```
WorkSheets.Add Count := 2, Before := Sheets(1)
```

Conjunto Range. Propiedades y Métodos. Ejemplos.

El conjunto Range representa una celda, un rango de celdas, una fila, columna sobre los cuales se puede actuar modificando sus propiedades mediante la acción de una gran variedad de métodos para devolver un objeto range.

Propiedad Range

Sintaxis

Range(Cadena).Método

Devuelve un objeto range que representa una celda o un rango de celdas.

El argumento *Cadena* representa a una celda o un rango de celdas. Por ejemplo A2, B5:B12, B2:M12.

Ejemplos simples:

```
Range("A1").Select                  ' Activa o selecciona la celda A1
Range("A1").Activate                'Activa o selecciona la celda A1
Range("A1").Value                   ' Devuelve el valor o contenido de la celda A1
Range("A1").Value = Expresión   ' Se asigna a la celda A1 el valor de Expresión.
Range("A1").Name = "Tasa"       ' Permite dar el nombre de Tasa a la celda A1
MsgBox Range("A1").Value        ' Permite visualizar el contenido de la celda A1
MsgBox Range("B3").Address()   ' Visualiza la celda B3
MsgBox Range("B3").AddressLocal()      ' Como en el caso anterior
Range("B3").AddComment "Esta es una celda de datos"
                                    ' Inserta en B3 el comentario que se indica
Range("B3").Clear                   ' Borra contenido y formato de la celda B3
Range("B3").ClearContents           ' Borra sólo el contenido de la celda B3
Range("B3").ClearComments      ' Elimina el comentario insertado en B3
Range("B3").ClearFormats            ' Borra solo el formato de B3
MsgBox Range("E3").Column      ' Emite el número de columna; en este caso 5.
MsgBox Range("J3:M21").Column          ' En este caso devuelve 10
Rango = "D2:F8"
```

```
MsgBox Range(Rango).Column              ' Devuelve 4
```

Veamos el siguiente grupo:

```
Rango = "D2:H15"

Range(Rango).Columns(3).Value = "M"

Range(Rango).Rows(3).Value = "MMMMMMM"
```

En este caso la variable Rango se define como el rango D2:H15; la siguiente sentencia asigna "M" a cada celda de la tercera columna de este rango; es decir a F3:F5

La tercera asigna el texto "MMMMMMM" a cada celda de la tercera fila; es decir a D4:H4

Las siguientes dos sentencias permiten copiar el contenido del rango B1:B6 de la Hoja2 hacia la Hoja3, a partir de la celda E5

```
MsgBox Range("B3").ColumnWidth        ' Devuelve el ancho de la columna B

Sheets("Hoja2").Range("B1:B6").Copy Destination:=Sheets("Hoja3").Range("E5")
```

La siguiente sentencia selecciona tres hojas del mismo libro (libro activo)

```
Worksheets(Array("Sheet1", "Sheet2", "Sheet4")).Select
```

En los siguientes ejemplos haremos uso de diversos métodos en el cual lo explicaremos explícitamente instrucción por instrucción.

Ejemplo 01

Asignar un texto a la celda B3

```
Sub Ej01()

'   Primera forma

Workbooks("Libro1").Worksheets(2).Range("B1") = "Hola Mundo B1 !!!"

'   Segunda forma

Worksheets(2).Range("B2") = "Hola Mundo B2 !!!"
```

```
'   Tercera forma
Worksheets(2).Activate
Range("B3") = "Hola Mundo B3 !!!"
'   Cuarta forma
Worksheets(2).Select
Range("B4").Select
Range("B4").Value = "Hola Mundo B4 !!!"
'   Quinta forma
Worksheets(2).Select
Range("B5").Formula = "Hola Mundo B5 !!!"
'   Sexta forma
Worksheets(2).Select
Range("B6").FormulaR1C1 = "Hola Mundo B6 !!!"
End Sub
```

Ejemplo 02

Ingresar datos desde el teclado hacia una celda

```
Sub Ej02
' Primera forma
XDat1 = InputBox("Ingresa un número")
Range("B1") = XDat1
' Segunda forma
Range("B1") = InputBox("Ingresa un número")
' Tercera forma
```

```
Cells(3, 2) = InputBox("Ingresa un número")

' Cuarta forma                Indicar en qué celda se guardará

Celda = InputBox("En que celda deseas almacenar el dato?")

XDat1 = InputBox("Ingresa el dato")

Range(Celda) = XDat1

'  Quinta forma

Celda = InputBox("Ingresa la celda donde se guardará")

Range(Celda) = InputBox("Ingresa el dato a ser guardado")

End Sub
```

Ejemplo 03

Obtener la suma de un rango de celdas de la hoja "Datos" y dejar el resultado en otra celda. Se recomienda grabar esta macro en un libro diferente al que se va a abrir.

En este ejemplo se usa el libro Ej03.xls y los datos de la hoja Totales. La macro se puede grabar en cualquier libro

```
Sub Ej03()

Libro = InputBox("Ingresa el nombre del libro")

Hoja = InputBox("Ingresa el nombre de la hoja")

'

' La siguiente instrucción abre la hoja del libro deseado

'

Workbooks.Open (Libro)

Sheets(Hoja).Activate

'

' La siguiente instruccioón ingresa una fórmula en F3
```

```
'
Range("F3") = "=Sum(B3:E3)"
'
' Como la celda active es aquella que contiene la fórmula, se copia
'
Selection.Copy
'
' Ahora se selecciona el rango destino y se pega en dicha selección
'
Range("F4:F14").Select
ActiveSheet.Paste
Application.CutCopyMode = False
'
' A continuación ingresa en B15 una formula, lo copia y lo pega en otro rango
'
Range("B15") = "=Sum(B3:B14)"
Range("B15").Copy
Range("C15:E15").Select
ActiveSheet.Paste
Application.CutCopyMode = False
End Sub
```


Ejemplo 04. Uso de nombres de rango

El siguiente ejemplo permite dar nombre a celdas individuales, ingresar una cantidad y luego almacenar en otra, una fórmula de cálculo. Define a C2 como "Tasa"; a D2 como "Capital" y a E2 como "Monto".

```
Sub Ej04()

' Activa la hoja 1

'

Sheets(1).Activate

'

' Ingresa constants en algunas celdas

'

Range("C2").Name = "Tasa"

Range("Tasa") = 0.19

Range("D2").Name = "Capital"

Range("E2").Name = "Monto"

'

' Pide ingresar un dato por teclado hacia una celda que tiene nombre de rango

'

Range("Capital") = InputBox("Ingrese el capital")

'

' Multiplica dos celdas por sus nombre de rango

'

Range("Monto") = "=Tasa*Capital"

End Sub
```

Ejemplo 05: Conversión de fechas

Ingresar una determinada fecha a una celda y luego obtener los nombres del mes y día de la semana

```
Sub Ej05
'
' Declaración de variable y arreglos de tipo cadena (String)
'
Dim Fecha  As String
Dim Ames(12) As String
Dim ADia(7) As String
'
' Se define constantes de tipo cadena. Debe tomar em cuenta los espacio em blanco ya que cada nombre de mês será manejado como texto de 9 caracteres.
'
TMes = "Enero    Febrero  Marzo    Abril    Mayo     Junio    Julio    Agosto   SetiembreOctubre  N
oviembreDiciembre"
TDias = "Lunes    Martes   MiercolesJueves   Viernes  Sábado   Domingo  "
'
' Asignación a um arreglo 9 caracteres para cada uno de los 12 elementos
' Lo mismo se hace com los días de la semana
'
For I = 1 To 12
    Ames(I) = Mid(TMes, 9 * (I - 1) + 1, 9)
Next
For I = 1 To 7
```

```
    ADia(I) = Mid(TDias, 9 * (I - 1) + 1, 9)
  Next
  '
  ' Se pide ingresar una fecha y se extrae por partes como valor numérico
  '
  Fecha = InputBox("Ingrese la fecha DD/MM/YYYY")
  Año = Val(Right(Fecha, 4))
  Mes = Val(Mid(Fecha, 4, 2))
  Dia = Val(Left(Fecha, 2))
  '
  ' Luego se emite los valores separados y convertidos a texto
  '
  Range("A2") = "La fecha ingresada es: "
  Range("B2") = Fecha
  DiaSem = Weekday(Cells(2, 2), 1)
  MsgBox "Corresponde al " & ADia(DiaSem) & Dia & " de " & Ames(Mes) & " del " & Año
  End Sub
```

Ejemplo 06. Ingresar apellidos y nombres y luego separarlos

El siguiente ejemplo permite ingresar los apellidos y nombres de una persona y luego los separa, almacenándolos en celdas del Excel.

```
Sub Eje06
Sub Apellidos_Y_Nombres()
'
```

```
' Se ingresa los datos hacia una variable de cadena
'
Texto = InputBox("Ingresa tus apellidos y nombres (Ap Am, Nombres)")
'
' Se ingresa constantes de cadena en la primera columna
'
Range("A1") = Texto
Range("A3") = "Ap. Paterno"
Range("A4") = "Ap. Materno"
Range("A5") = "Nombres"
'
' Se extrae cada uno de los apellidos y nombres a partir del dato ingresado
'
Range("B3") = Left(Texto, InStr(Texto, " "))
Range("B4") = Mid(Texto, InStr(Texto, " "), Len(Texto) - InStr(Texto, ",") - 6)
Range("B5") = Right(Texto, Len(Texto) - InStr(Texto, ","))
End Sub
```

Ejemplo 07. Almacenar n datos en un rango de celdas ingresados por teclado

```
Sub Ej07()
'
N = InputBox("Nro. de datos")
CeldaIn = InputBox("A partir de qué celda (Ej: B2, M12)?")
'
```

```
' Primero convertimos la fila y columna en variables
'
' La función Ucase convierte a mayúscula el carácter
' La función ASC devuelve el valor ASCII del carácter
' La función VAL convierte a valor la cadena que contiene números
'
Select Case Len(CeldaIn)
   Case 2: Y = Val(Asc(UCase(Left(CeldaIn, 1))) - 64)
           X = Val(Right(CeldaIn, 1))
   Case 3: Y = Val(Asc(UCase(Left(CeldaIn, 1))) - 64)
           X = Val(Right(CeldaIn, 2))
End Select
'
' Ahora ingresamos los datos usando Cells(x,y)
'
For I = 1 To N
   Cells(X + I - 1, Y) = InputBox("Ingrese el dato: " & I)
Next
End Sub
```

Ejemplo 08. Ingresar pareja de datos, separados por ",", hacia dos columnas.

```
Sub Ej08()
Dim StDato As String
' Usaremos la columna A y B a partir de la fila 2
```

```
'
Row = 2
Col = 1
Range("A1") = "X"
Range("B1") = "Y"
'
' La siguiente sentencia permite centrar el dato contenido en el rango
'
Range("A1:B1").HorizontalAlignment = xlCenter
'
' Ingreso de datos
'
nDat = InputBox("Número de datos")
For I = 1 To nDat
   Dato = InputBox("Dato: " & I)
   Cells(I + 1, 1) = Val(Mid(Dato, 1, InStr(Dato, ",") - 1))
   Cells(I + 1, 2) = Val(Mid(Dato, InStr(Dato, ",") + 1, Len(Dato) - InStr(Dato, ",")))
Next
End Sub
```

Ejemplo 09. Usar el módulo anterior para realizar otros cálculos

Usando el módulo anterior, obtener en las siguientes columnas, los cálculos de X*X, X*Y, Y*Y y X^2*Y. Para luego obtener las sumatorias en la última fila.

Procedimiento principal:

Paso 1: Declaración de variables y arreglos

Paso 2: Llamada al módulo anterior de lectura e ingreso de datos

Paso 3: Llamada al módulo de cálculo y almacenamiento en columnas

A continuación la única instrucción global que debe codificarse así como los módulos adicionales al módulo principal.

Hemos vuelto a copiar el módulo anterior Ej08()

```
Dim nDat As Variant

Sub PMain()

'

'     Nombre de columna

'

Range("C1") = "X²"

Range("D1") = "X*Y"

Range("E1") = "X²*Y"

Range("F1") = "Y²"

Range("G1") = "X*Y²"

'   Inicialización de variables

'   Llama a ingreso de datos

'

Ej08    ' Ingreso de datos

'

'   Obtiene el número de datos invocando el procedimiento NroDatos.

'   Se encuentra en la siguiente página

NroDatos

'
```

```
' La variable global (común a todos los procedimientos) se reduce en 1
'
nDat = nDat - 1
'
'   Loop para calcular las columnas
'
For I = 1 To nDat
    Cells(I + 1, 3) = Cells(I + 1, 1) ^ 2
    Cells(I + 1, 4) = Cells(I + 1, 1) * Cells(I + 1, 2)
    Cells(I + 1, 5) = Cells(I + 1, 3) * Cells(I + 1, 2)
    Cells(I + 1, 6) = Cells(I + 1, 2) ^ 2
    Cells(I + 1, 7) = Cells(I + 1, 1) * Cells(I + 1, 6)
Next
'   Obtiene la suma de todas las columnas y los guarda una fila más abajo
'
' VarSuma = Range(Cells(2, 1), Cells(21, 1)): Ejemplo de cómo seleccionar un
' determinado rango, usado líneas abajo
For I = 1 To 7
    Cells(2, I).Select
' Range("A2").Select
' Selecciona el rango de datos de la i-ésima columna
'
    VarSuma = Range(Selection, Selection.End(xlDown))
```

```
'Suma el rango seleccionado y lo almacena
    Cells(nDat + 3, I) = Application.WorksheetFunction.Sum(VarSuma)
Next
End Sub
Sub Ej08()
Dim StDato As String
' Usaremos la columna A y B a partir de la fila 2
'
Row = 2
Col = 1
Range("A1") = "X"
Range("B1") = "Y"
'
' La siguiente sentencia permite centrar el dato contenido en el rango
'
Range("A1:B1").HorizontalAlignment = xlCenter
'
' Ingreso de datos
'
nDat = InputBox("Número de datos")
For I = 1 To nDat
    Dato = InputBox("Dato: " & I)
    Cells(I + 1, 1) = Val(Mid(Dato, 1, InStr(Dato, ",") - 1))
```

```
    Cells(I + 1, 2) = Val(Mid(Dato, InStr(Dato, ",") + 1, Len(Dato) - InStr(Dato, ",")))
Next
End Sub
Sub NroDatos()
nDat = Columns("A:A").Range("A65536").End(xlUp).Row
End Sub
```

Nota 1:

Este es un ejemplo independiente del módulo principal usado para probar la definición de un rango y la forma de cómo sumar dicho rango

```
Sub Macro4()
' VarSuma = Range(Cells(2, 1), Cells(21, 1))
For I = 1 To 7
    Cells(2, I).Select
' Range("A2").Select
    VarSuma = Range(Selection, Selection.End(xlDown))
'Sumar el rango
    Cells(24, I) = Application.WorksheetFunction.Sum(VarSuma)
Next
End Sub
```

Nota 2:

Si abre el archivo **Uso de Módulos.xls**, en la primera hoja encontrará un botón al cual se le ha asignado la macro PMain(). De manera que si desea ver una demostración de esto, debe hacer clic en dicho control.

Ejercicios

a) Cómo diferencia una variable local de una pública?

Las variables locales son aquellas que se encuentran activas dentro de un procedimiento o módulo; esto significa que fuera del procedimiento la variable no existe. El ámbito de estas variables es dentro del procedimiento en el cual se las define. Estas variables se las declara mediante la sentencia DIM.

Ejemplos:

DIM XIngreso As Double

DIM Codigo As variant

Las variables públicas son aquellas cuya definición y valor se encuentran activas en todos los procedimientos o módulos que conforman un proyecto. Se las define mediante la sentencia PUBLIC y se las define antes del primer módulo y fuera de él.

Ejemplo

PUBLIC Saldo As Double

PUBLIC Precio, Stock, Tasa As Double

b) Qué hacer si la ejecución de una instrucción del procedimiento falla; es decir, cuando el programa encuentra un error?

El manejo de errores en un procedimiento puede ser administrado de varias formas, para los cuales existen ciertas sentencias especiales que se colocan en la cabecera del procedimiento:

i) On Error Resume Next:

Significa que si al ejecutarse el procedimiento se produce un error, la ejecución debe continuar con la siguiente sentencia del procedimiento.

ii) On Error GoTo Labxx

Si se produce un error en la ejecución, la ejecución continuará en la sentencia cuya etiqueta es Labxx (este es un nombre cualquiera).

```
Sub Decode()

On Error GoTo Labxx

Sentencia - 1

Sentencia – 2.

…….

Labxx: Sentencia – kk

End Sub
```

iii) On Error GoTo 0

Permite restablecer el manejo de errores, después de haber usado una de las dos alternativas anteriores.

c) Obtenga el nombre del libro activo

```
Sub LibroName()

DIM LibNom As String

LibNom = WorkBooks(1).Name

MsgBox "El nombre del libro es: " & LibNom

End Sub
```

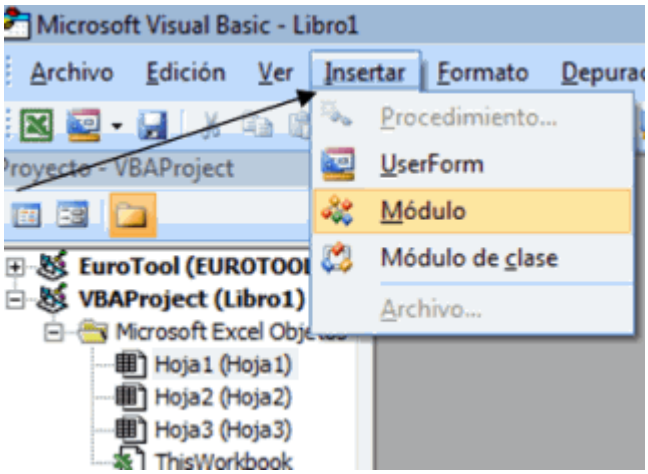
d) ¿Cuál es el nombre del primer libro abierto y el del libro activo?

```
Sub LibNa()

Dim Libro01, LibroAct As String

Libro01 = Workbooks(1).Name

LibroAct = ActiveWorkbook.Name

MsgBox ("El libro 1: " & Libro01 & Chr(10) & "El Libro activo: " & LibroAct)

End Sub
```

Unidad 6. Creación de formularios de usuario

Ejemplos de interacción con módulos y macros.

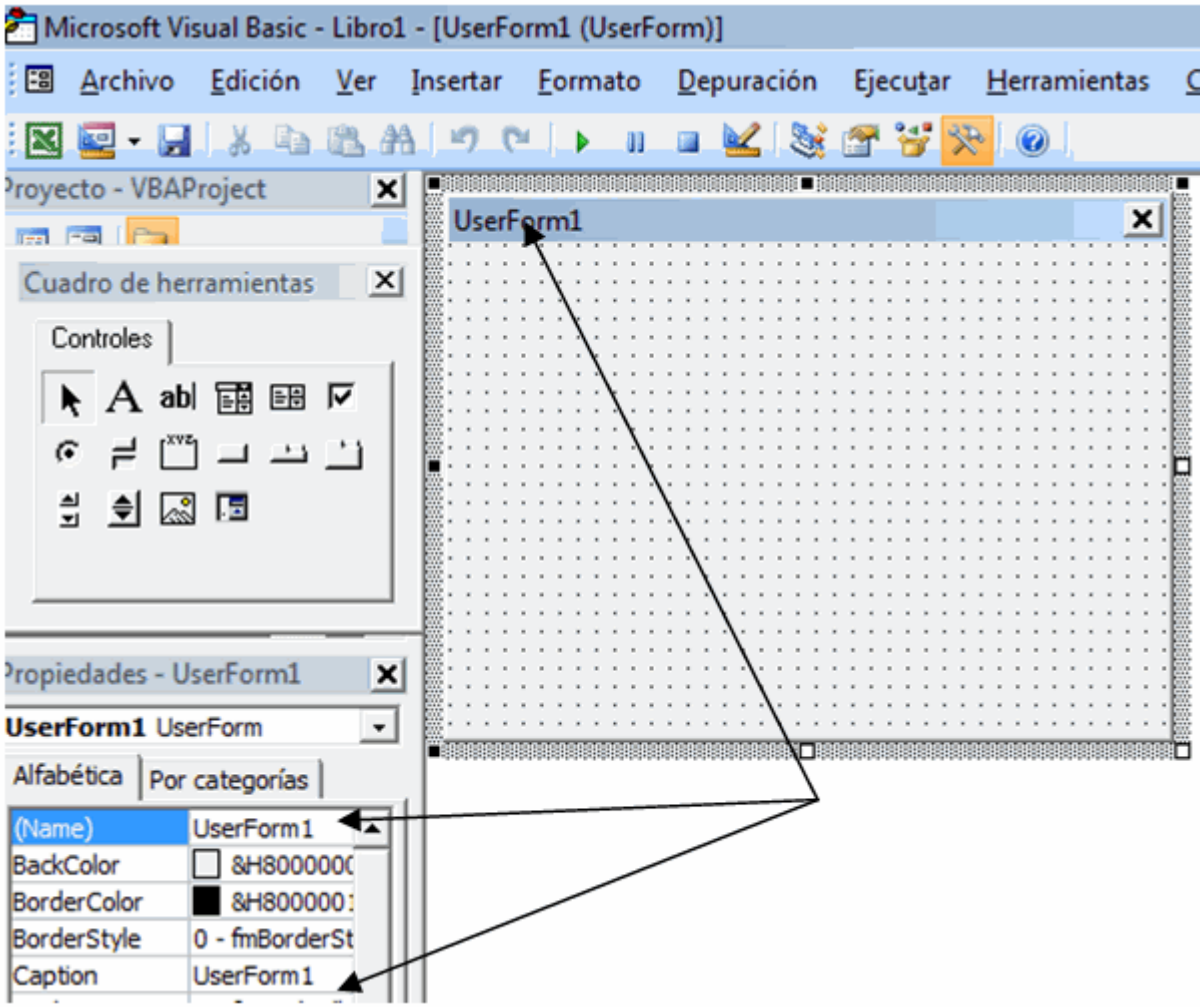
Un formulario es una ventana o cuadro de diálogo que contiene un conjunto de controles insertados por nosotros desde la barra de herramientas o cuadro de control al cual se le denomina también conjunto de Controles Activex.



En el VBA a un formulario se le denomina USERFORM quizás recogiendo el nombre que tradicionalmente lo ha usado el lenguaje de programación Visual Basic.

Use la siguiente secuencia para insertar un formulario en la ventana del editor del Visual Basic.

La siguiente es una imagen de un UserForm

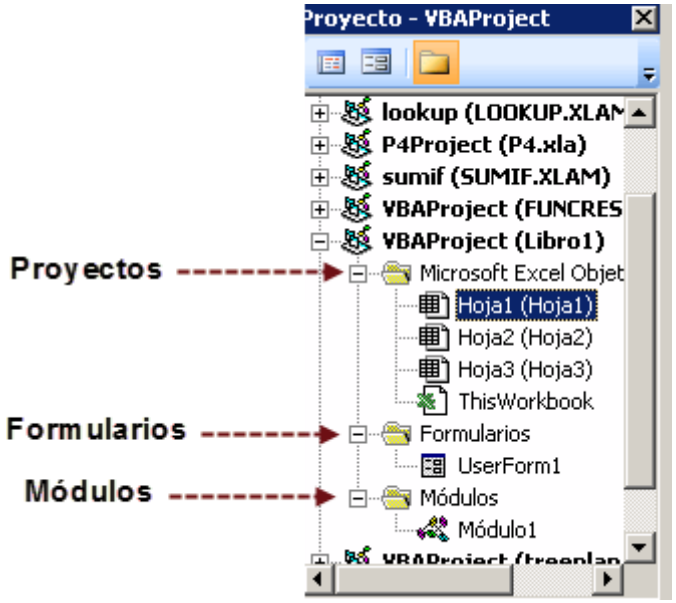


En la figura anterior se muestra el formulario llamado UserForm1. Si se desea cambiar este nombre se debe hacer clic en la opción Name de la ventana de propiedades. Si se desea que el formulario se llame "Panel de ingreso de datos", entonces se debe hacer clic en la propiedad Caption de la ventana de propiedades y digitar dicho texto en el lado derecho de esta propiedad.

Ventana de Proyectos

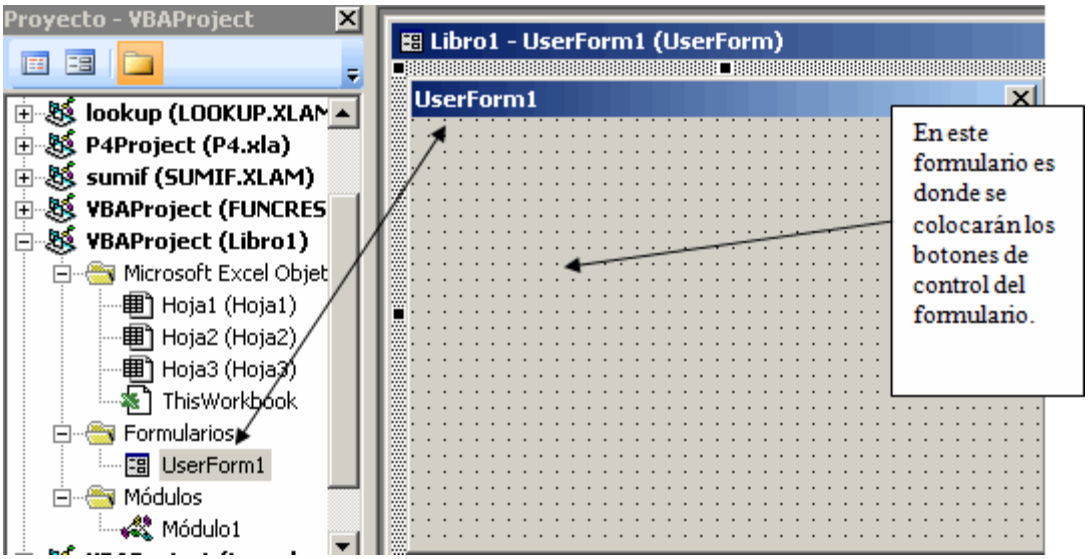
Al abrir el Editor del VBA Usando <Alt> + <F11> se acede a un conjunto de ventanas que conforman la ventana del editor.

Entre estas ventanas tenemos:

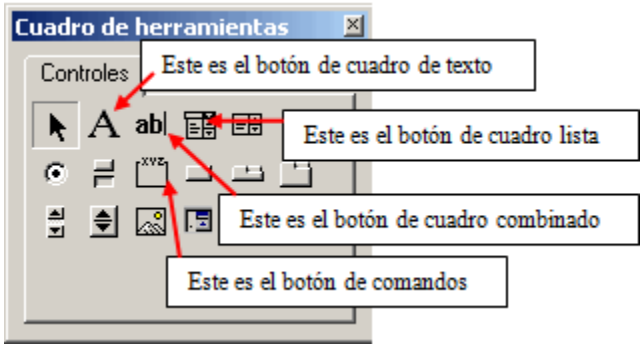


Si en esta ventana no existe ningún formulario, se debe usar <Insertar> - <UserForm>. Esto permite crear la carpeta <Formularios> y dentro de ella estarán cada uno de los formularios que se hayan insertado. En el caso de la figura anterior, se ha creado un primer formulario llamado Userform1.

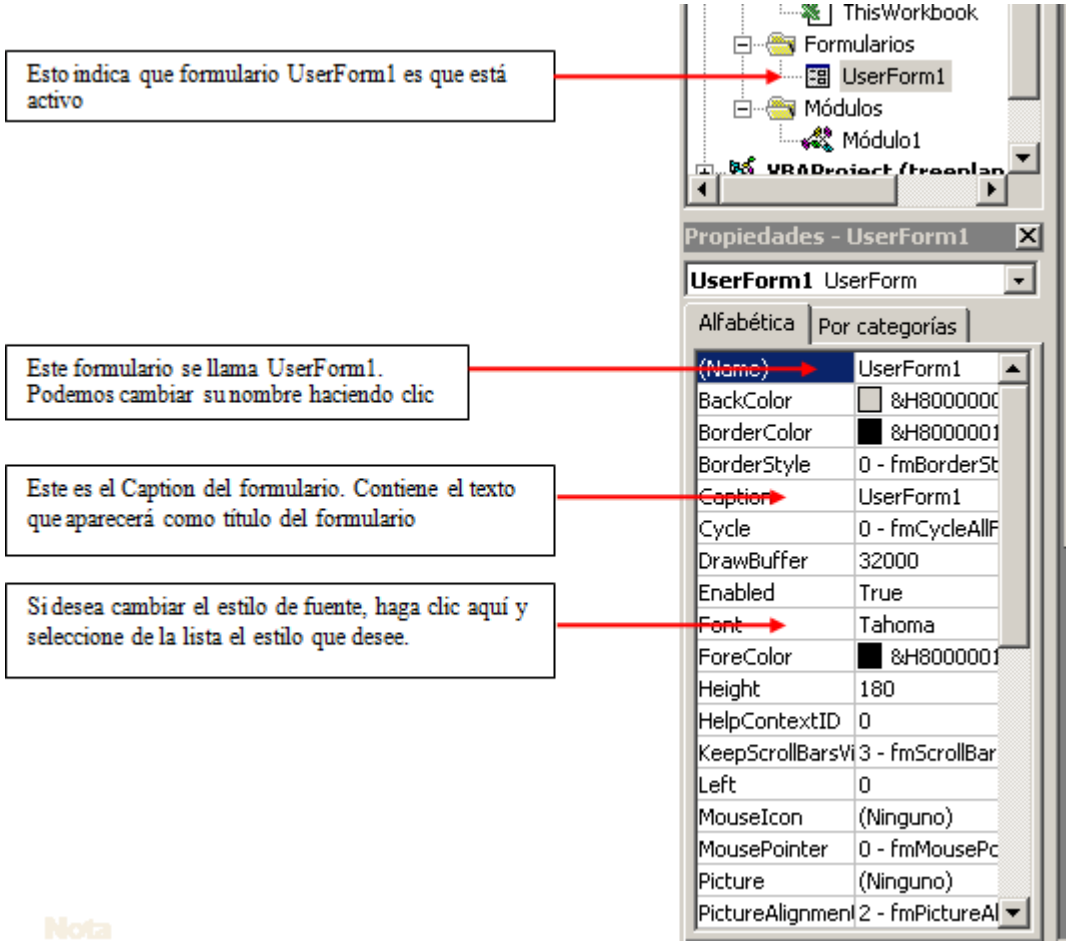
Al insertar un formulario de usuario en la ventana central se dispondrá del formulario mismo, según se muestra en la siguiente figura.



Al insertar un formulario, se dispondrá del cuadro de herramientas compuesto por un conjunto de botones de control del formulario, los que se muestra en la siguiente figura.



Por debajo de la ventana de proyectos se dispondrá de la Ventana de Propiedades. En esta ventana se visualizará las propiedades que posee un determinado objeto. En la siguiente imagen se muestra la ventana de propiedades de un formulario, ya que éste es el que está activo.



Nota

Si se hace clic en una hoja de un libro, en la ventana de

Propiedades se visualizará las propiedades de la hoja.

Cómo ejecutar un formulario

En la barra de herramientas que aparece debajo de los comandos, se muestra tres controles similares a  . Para ejecutar un formulario es suficiente hacer clic en el primer botón. Para detener la ejecución del formulario, se debe hacer clic en el segundo y el tercero permite dar por terminado o cancelar la ejecución del formulario.

El contenido de un formulario depende de qué queremos hacer mediante esta ventana:

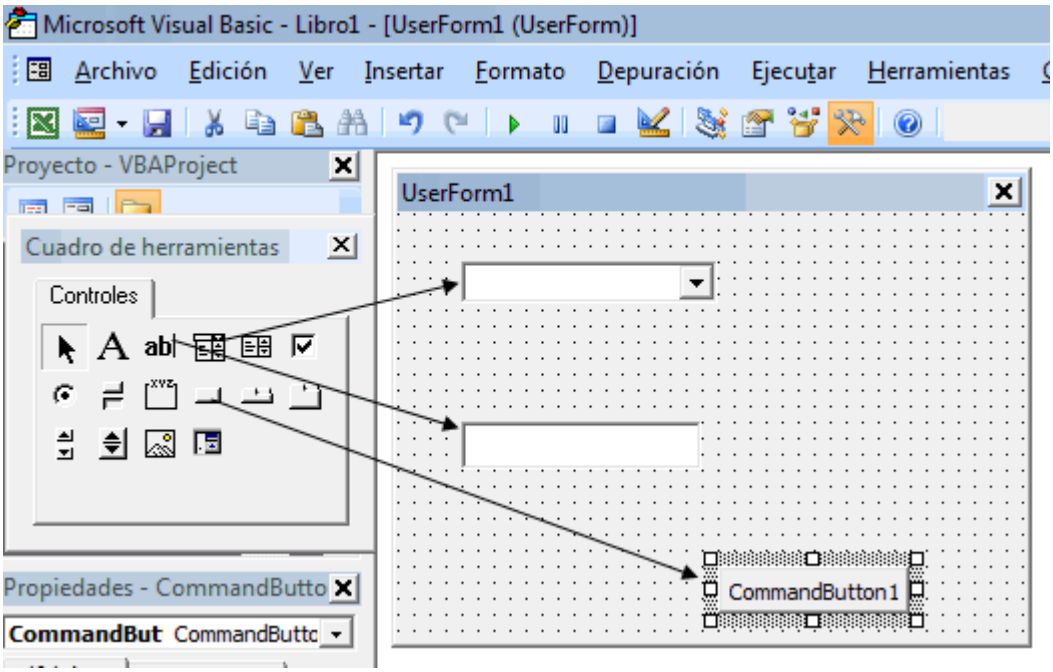
Puede servirnos como panel de ingreso de datos y emisión de resultado de cálculo que hagamos con los datos

Puede servirnos para ingresar datos a una hoja de cálculo

Puede servirnos para recuperar datos almacenados en una hoja de cálculo

Puede servirnos como un panel de diálogo que permita extraer y/o ingresar datos a y desde una hoja de un libro.

En la siguiente figura hemos insertado un cuadro combinado, un cuadro de texto y un botón de comando.



En esta figura, se encuentra seleccionado el botón de comando.

Cada uno de estos controles se han insertado haciendo clic en sus iconos que aparecen en la ventana del Cuadro de herramientas y luego trazando un pequeño recuadro en el formulario.

Cada uno de estos controles tienen un nombre interno, el cual se modifica usando la propiedad **Name**, el cual es usado en la codificación de las acciones que se debe realizar al activar dicho control. Alguno de ellos tienen también la propiedad **Caption**, que permite ingresar un texto en dicho botón de control y que servirá para colocarle una

etiqueta al botón, como ocurre con el botón de comando que se muestra en la figura anterior: CommandButton 1.

Si se hace doble clic en uno de estos controles, se puede acceder directamente al código que controla el uso de dicho botón.

Las acciones que se han enumerado líneas arriba se pueden llevar a cabo mediante el uso de los cuadros de control.

Cuadros de control del UserForm

A continuación pasaremos a describir los botones de control más comunes aplicándolo a ejemplos sencillos.

Botón de comando

Este control permite realizar una determinada acción; puede servir para abrir un nuevo formulario, para asignar valores iniciales, para modificar alguna propiedad de otros controles, para transferir datos a y desde una hoja de trabajo. Puede servir también para dar por terminado el uso del formulario.

El evento que se genera al presionarlo es el **evento clic**.

Haga clic en el botón de comando y luego trace un pequeño rectángulo en la parte inferior derecha del formulario denominado **UserForm1**.

Al hacer doble clic sobre este control, se despliega el siguiente módulo:

Private Sub CmdFin_Click()

End Sub

Al interior de este módulo se colocarán todas las instrucciones que se desea realizar al hacer clic en este botón.

Algunas de sus propiedades son:

Name: Propiedad usada para definir el nombre del objeto cuadro de texto

AutoSize: Permite que el tamaño del objeto se modifique en función al contenido

BackColor: Permite seleccionar un color de fondo de dicho cuadro de texto

Caption: Permite ingresar un determinado texto el cual aparecerá en el botón.

TabIndex: Sirve para ingresar un determinado número a fin de ordenar los cuadros de control usados en el formulario.

Ejemplo: Insertar un control para dar por terminado un formulario.

Active la Ventana de propiedades (si no lo estuviera) presionando la tecla <F4> o usando la secuencia <Ver> - <Ventana de propiedades>.

Estando seleccionado el botón insertado, ubique la propiedad (Name), que es la primera. Haga clic en ella y digite **CmdFin**.

Ahora haga clic en la propiedad Caption y digite **Terminar**.

Ejemplo 01

Crear un formulario que se llame Forma01, que su caption sea "Este es un formulario"; que el color de fondo sea verde claro.

Procedimiento:

P1. Abra el editor usando <Alt>+F11. Insertar un formulario usando <Insertar> - <UserForm>.

P2. Haga clic en (Name) y luego digite Forma01

P3. Haga clic en <Caption> y luego digite "Este es un formulario"

P4. Haga clic en <Back color> y desplegando la lista seleccione el color verde claro en <Paleta>.

Ejemplo 02

Crear un formulario que se llame Forma02, que tenga un botón de comando y que éste permita dar por terminado el uso del formulario. Que su caption sea Botón de comando. Luego al hacer clic en dicho botón, durante la ejecución del formulario, debe dar por terminado su uso.

Procedimiento:

P1. Inserte un nuevo UserForm.

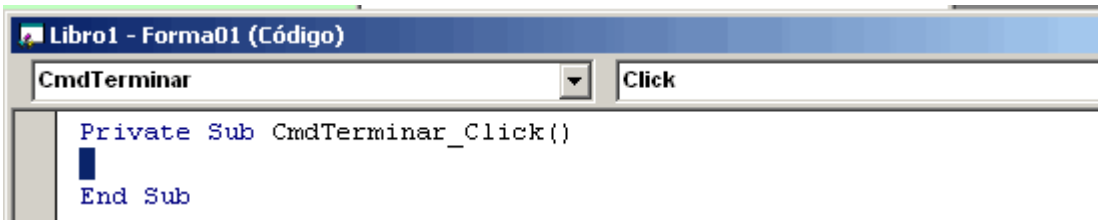
P2. Haga clic en (Name) y digite Forma02. Haga clic en <Caption> y digite "Botón de comando".

P3. Haga clic en el Botón <Botón de comando> del cuadro de herramientas. Luego, en el formulario, trace un pequeño rectángulo en la parte inferior derecha.

P4. Haga clic en (Name) y digite **CmdTerminar**. En <Caption> digite **Terminar**.

P5. Ahora vamos a insertar el código que permita dar por terminado la ejecución del formulario.

Para ello haga doble clic en el botón que ha insertado

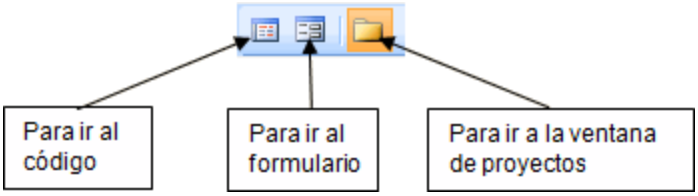


Como verá se inserta el siguiente segmento de código:

Dentro de este procedimiento digite **End**

Nota:

Para volver al formulario haga clic en



Para ejecutar el formulario haga clic en el icono  de la barra de herramienta Estándar.

Al estar en modo de ejecución, haga clic en el botón <Terminar> y verá que el formulario pasa a modo de edición en la ventana del editor.

Ejemplo 03

Crear un formulario que al hacer clic en un botón de comando, se muestre en una ventana del MsgBox, la ruta del libro que esté en uso y al hacer clic en el segundo, se dé por terminado la ejecución del formulario.

Procedimiento

P1. Insertar un UserForm que se llame Forma03

P2. Insertar dos botones de comando.

Botón 1: Nombre: CmdRuta Caption: Devuelve la ruta

Botón 2: Nombre: CmdFin Caption: Terminar

P3. Luego de hacer doble clic en el botón 1, ingrese el siguiente código:

```
Dim Ruta As Variant
Workbooks(1).Activate
Ruta = ActiveWorkbook.Path
MsgBox Ruta
```

P4. Luego de hacer clic en el botón 2, ingrese el siguiente código:

```
End
```

P5. Ejecute el formulario haciendo clic en el icono 

Ejemplo 04

Ahora diseñe un formulario que permita ingresar por teclado la ruta en donde se encuentra los archivos que se va a usar, que pida el nombre del libro a ser abierto y luego muestre el número de hojas de dicho libro.

Procedimiento:

P1. Inserte un nuevo formulario. Haga que su nombre sea FrmAbrir; que su caption sea "Abrir libros".

P2. Inserte dos botones de comando. Que el primero tenga por Name: CmdAbrirLibro y su caption sea "Abre Libro". Que el segundo botón tenga por Name: CmdTerminar y su caption sea "Terminar".

P3. Luego de hacer doble clic en el primer botón, ingrese las siguientes líneas de código:

```
Dim Ruta As Variant
Dim LibroXx As Variant
```

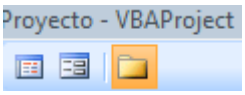
Ruta = InputBox("Ingrese la ruta en donde se" & Chr(10) & "encuentren los archivos a ser usados") + "/"

LibroName = InputBox("Nombre del libro")

Libro = Trim(Ruta) + Trim(LibroName) + ".**xls**"

Workbooks.Open Libro

MsgBox (Workbooks.Application.ActiveWorkbook.Sheets.Count)

P4. Vuelva al formulario haciendo clic en el botón central de 

Y a continuación haga doble clic en el segundo botón para digitar la instrucción End.

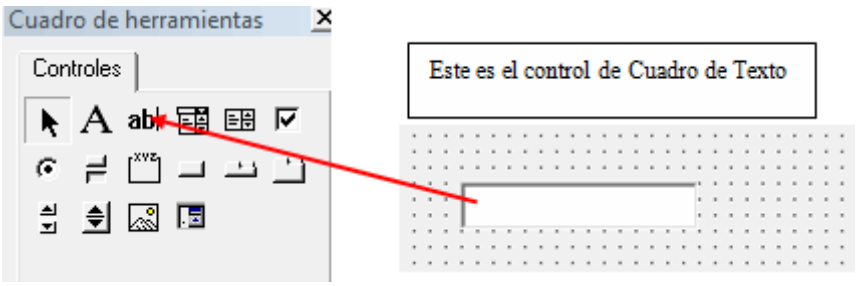
P5. Supongamos que desea abrir el archivo Tempo.xls, que se encuentra en la ruta: d:\Mis archivos\Libros. Ejecute el formulario. Ingrese la ruta; haga clic en <Aceptar> y digite el nombre del archivo sin extensión que desea abrir.

Nota:

El libro a ser abierto debe ser de extensión xls. Si la extensión es otra será suficiente modificar el código.

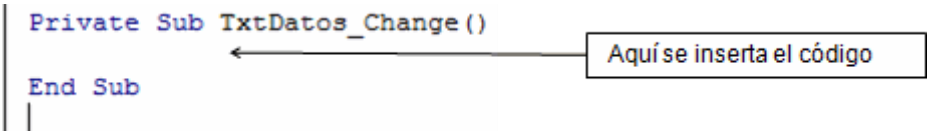
Cuadro de Texto

Haga clic en el icono  y luego trace un rectángulo como se muestra.



El control de cuadro de texto se usa para ingresar algún dato por teclado o recibir algún resultado producto de algún cálculo o proveniente de alguna celda de la hoja que estuviera activa.

Si se hace doble clic en un cuadro de texto, se despliega el procedimiento que le corresponde el cual contendrá el código que se ejecuta o en que se puede insertar el código que queremos que se ejecute sea cuando su contenido se modifique.



Algunas de sus propiedades más relevantes son:

Name: Propiedad usada para definir el nombre del objeto cuadro de texto

BackColor: Permite seleccionar un color de fondo de dicho cuadro de texto

Enabled: Permite activar o no el botón sin necesidad de hacer clic en él.

TabIndex: Sirve para ingresar un determinado número a fin de ordenar los cuadros de control usados en el formulario.

Text: Permite asignar en forma predeterminada una cadena de texto.

Value: Como en el caso anterior, se puede asignar determinado valor.

Ejemplo 05

Diseñe un formulario que contengo dos cuadros de texto y un botón de comando. El primer cuadro de texto debe permitir ingresar un texto cualquiera y mientras se digite, simultáneamente se debe emitir el mismo texto en el segundo cuadro de texto. El botón de comando debe dar por terminado el uso del formulario.

Procedimiento

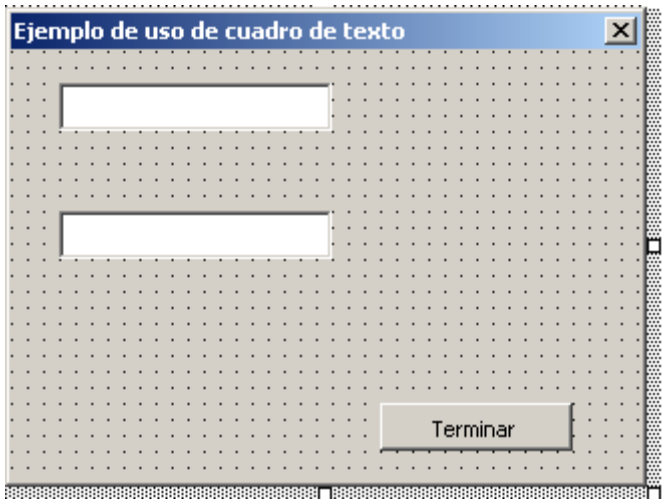
P1. Inserte un nuevo formulario

P2. Inserte en este formulario dos cuadros de texto y un botón de comando

P3. Defina las propiedades de cada botón según los siguientes datos:

Botón	Name	Caption
Formulario	FrmPanel01	Ejemplo de uso de cuadro de texto
Cuadro de texto 1	TxtDato	
Cuadro de texto 2	TxtSalida	

Botón de comando	CmdFin	Terminar



P4. Debemos hacer que TxtDatos sea habilitado para ingresar datos. Para ello haga doble clic en una parte del formulario. Se mostrará el siguiente procedimiento:

Private Sub UserForm_Click()

End Sub

Aquí ingresaremos el código:

TxtDato.Enabled = True

A fin de habilitar el primer cuadro de texto para ingresar un dato cualquiera.

P5. Haciendo doble clic en el segundo cuadro de texto y digite:

TxtSalida.Text = TxtDato.Text

P6. Haga doble clic en el botón de comando e ingrese el código: **End**

P7. Vuelva al formulario y haga clic en el icono para ejecutarlo. Ahora ingrese un texto bastante largo en el primer cuadro de texto y observe lo que sucede en el segundo cuadro de texto. Finalmente haga clic en <Terminar> para finalizar la ejecución del formulario.

Etiqueta

Este control permite ingresar un texto a fin de que se pueda usar como un comentario o como una etiqueta que acompañe a un cuadro de texto o algún otro botón de control.

Las propiedades de mayor uso son:

Name: Para darle nombre al comentario, aunque no es de importancia

Caption: Esta es el más importante pues en ella se ingresa o aparece de manera predeterminada, el mensaje que queremos transmita dicho control.

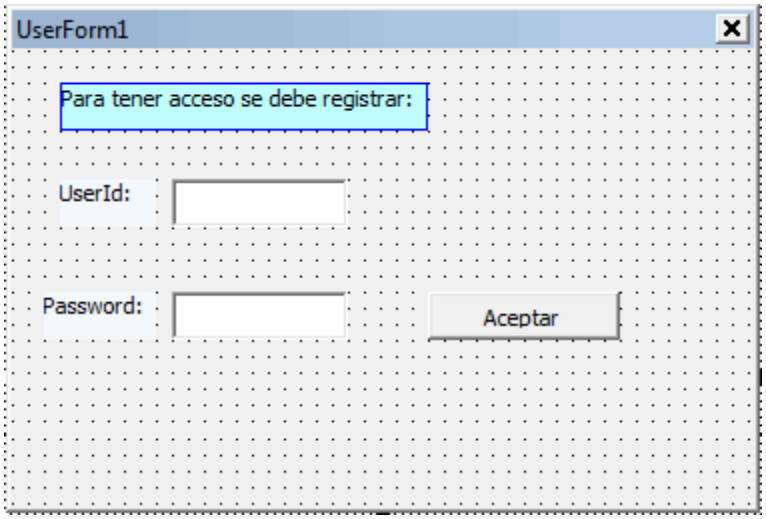
Así como en los anteriores, se puede cambiar el tamaño, se puede definir el color del borde, del fondo de dicho control.

Ejemplo 06: Aplicación del Ejemplo 05 para Autorización de acceso

Diseñe un formulario que solicite el código de usuario (UserId) y la contraseña (Password) para simular el acceso a un archivo o sistema. Al ingresar la contraseña, en el cuadro de texto que le corresponda se debe imprimir un asterisco por cada carácter de la contraseña.

Procedimiento:

Inserte un nuevo formulario (userform) y que tenga el siguiente diseño:



A continuación defina cada uno de los controles según la siguiente tabla:

Botón	Name	Caption
Formulario	FrmPanel02	Panel de acceso
Etiqueta 1	Lbl01	Para tener acceso se debe registrar;
Etiqueta 2	Lbl02	UserId:
Etiqueta 3	Lbl03	Password:
Cuadro de texto 1	TxtUserId	

Cuadro de texto 2	TxtPassword	
Botón de comando	CmdFin	Terminar

Al hacer doble clic en cada uno de los controles debe ingresar el código que se muestra a continuación:

```
Private Sub CmdAceptar_Click()

MsgBox ("Su usuario es: " + txtUserId.Text & Chr(10) & "Su contraseña es: " + TxtPassword.Text)

End

End Sub

Private Sub TxtPassword_Change()

TxtPassword.PasswordChar = "*"

End Sub

Private Sub txtUserId_Change()

TxtPassword.Enabled = True

End Sub

Private Sub UserForm_Click()

TxtUserId.Enabled = True

End Sub
```

Ahora ejecute el formulario y después de ingresar un texto cualquiera para el código de usuario y la contraseña, haga clic en <Aceptar>.

1.1. Cuadro combinado

El control de cuadro combinado permite insertar al interior de él una lista de elementos los cuales pueden ser seleccionados y realizar otras acciones a partir de ellas.

Los elementos que forman parte de la lista pueden ser añadidos mediante peticiones por teclado a mediante la descarga de otra tabla o un rango de celdas de una hoja de cálculo.

Ejemplo 07

Diseñe un formulario que contenga un cuadro combinado y dos botones de comandos. Que el cuadro combinado tenga por nombre CboLista; un botón de comando se llame CmdActivar y su caption sea <Activar> y el segundo botón de comando tenga por nombre CmdFin y su caption sea <Terminar>. Al hacer doble clic en el botón Activar debe ingresar el siguiente código:

El código que se debe insertar en cada uno de estos botones de control es la siguiente:

```
Private Sub CmdActvar_Click()

CboLista.AddItem "Estadística General"

CboLista.AddItem "Estadística Aplicada I"

CboLista.AddItem "Estadística Aplicada II"

CboLista.AddItem "Análisis Mutivariado"

CboLista.AddItem "Matemática Básica I"

CboLista.AddItem "Matemática para los Negocios"

CboLista.AddItem "Ecuaciones generales"

CboLista.AddItem "Teoría del crecimiento"

CboLista.AddItem "Creación de modelos económicos"

End Sub

Private Sub CmdFin_Click()

End

End Sub

Private Sub UserForm_Click()
```

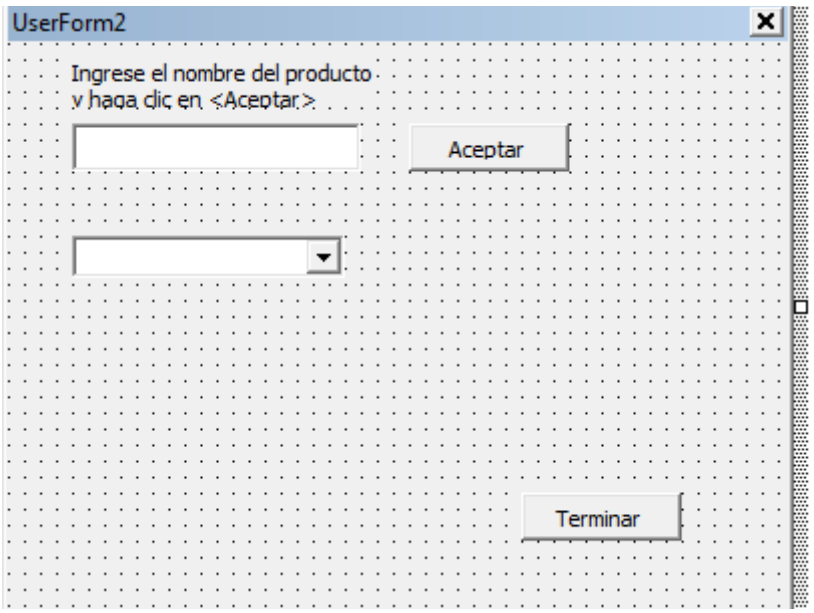
```
End Sub
```

Ejemplo 08

Diseñe un formulario que permita ingresar una lista de elementos hacia un cuadro combinado y que son digitados en un cuadro de texto. Cada vez que se digita una línea, se debe hacer clic en un botón de comando para ingresar el dato a la lista. Al final se debe hacer clic en un segundo botón de comando para terminar la ejecución del formulario.

Procedimiento:

La siguiente imagen muestra los elementos que debe tener el formulario.



La tabla siguiente muestra el nombre de los elementos

Botón	Name	Caption
Etiqueta	Lbl01	Ingrese el nombre del producto y haga clic en <Aceptar>
Cuadro de texto	TxtElemento	
Botón de comando 1	CmdAceptar	Aceptar
Botón de comando 2	CmdFin	Terminar

El código de cada control es el siguiente:

```
Private Sub CmdAceptar_Click()

CboLista.AddItem TxtElemento.Text
```

```
TxtElemento.Text = ""

TxtElemento.SetFocus

End Sub

Private Sub CmdFin_Click()

End

End Sub

Private Sub UserForm_Click()

TxtElemento.SetFocus

End Sub
```

Al ejecutar el formulario, cada vez que ingresa un nombre o texto, debe hace clic en <Aceptar>. En cualquier momento puede desplegar la lista en el cuadro combinado para ver los elementos que se van añadiendo. Para terminar debe hacer clic en <Terminar>.

1.2. Cuadro de Lista

Como el control Cuadro combinado, el cuadro de lista permite almacenar una lista de elementos a fin de seleccionar uno o más elementos y extraer una copia de ellos.

Los elementos a ser almacenados pueden ser ingresados por teclado, desde un procedimiento o desde una hoja de cálculo del Excel.

Nota:

Tanto en el cuadro combinado como en el cuadro de lista, los elementos contenidos se hacen disponible cuando se produce algún evento de cambio.

Ejemplo 09

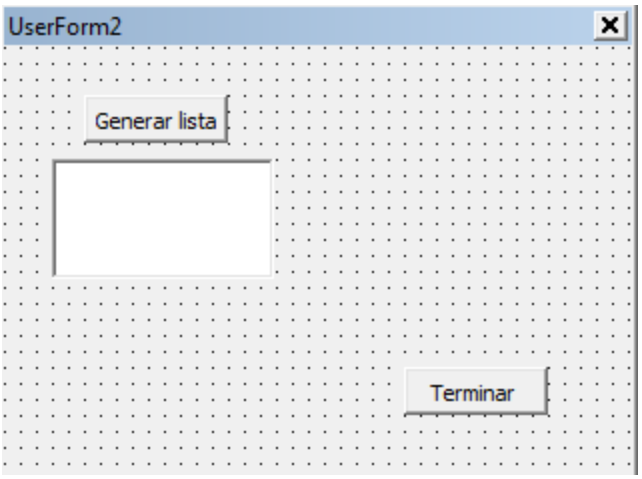
Inserte un formulario y dentro de él, un cuadro de lista a fin colocar una lista de nombres al hacer clic en un botón de comando.

Procedimiento:

Luego de insertar un UserForm, coloque dentro de él los siguientes controles:

Control	Nombre, caption
Comando	CmdLista, Generar Lista
Comando	CmdFin, terminar
Cuadro de Lista	LstLista

La siguiente imagen muestra estos controles sobre un formulario y los procedimientos respectivos.




```
Private Sub CmdLista_Click()

LstLista.AddItem "Carlos"

LstLista.AddItem "Salomé"

LstLista.AddItem "César"

LstLista.AddItem "Iván"

LstLista.AddItem "Pedro"

LstLista.AddItem "Miguel"

LstLista.AddItem "Janet"

LstLista.AddItem "Karla"

End Sub

Private Sub CmdFin_Click()
```

```
End

End Sub
```

Al ejecutar este formulario primero debe hacer clic en <Generar lista>. Para terminar, haga clic en <Terminar>.

Ejemplo 10

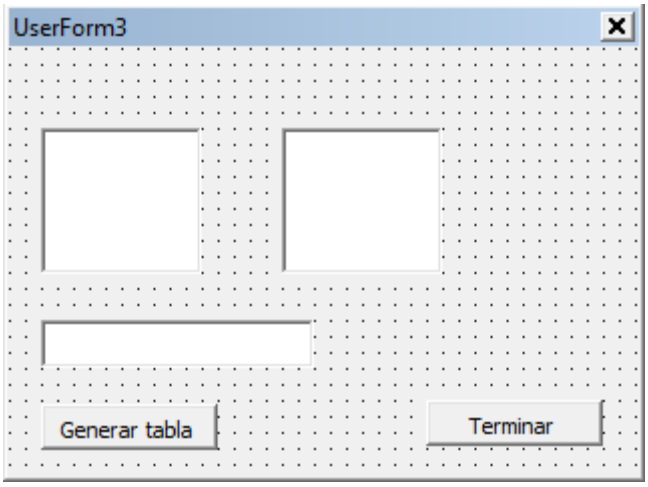
El siguiente ejemplo permite colocar en dos listas los nombres de los meses y los días de semana. Luego, extrae el número de mes, día y año de la fecha actual, para visualizar los respectivos nombres en un cuadro de texto.

Procedimiento:

Luego de insertar un UserForm, inserte y defina los siguientes cuadros de control:

Control	Nombre, caption
Comando	CmdGenera, Generar tabla
Comando	CmdFin, Terminar
Cuadro de Lista	LstMeses
Cuadro de Lista	LstDias
Cuadro de texto	TxtFecha

Una muestra del formulario es la siguiente imagen



A continuación listamos el código:

```
Private Sub CmdFin_Click()

End

End Sub

Private Sub CmdGenera_Click()

Dim Fecha As Variant

Dim Mes(12) As Variant

Dim Dia(7) As Variant

'  Meses y Dias son cadenas de texto a partir del cual extraeremos los nombres

'  de los meses y los días

Meses = "Enero  Febrero Marzo  Abril  Mayo  Junio  Julio  Agosto SetiembreOctubre N
oviembreDiciembre"

Dias = "Lunes   Martes  MiercolesJueves  Viernes Sábado  Domingo  "

' A continuación se extrae 9 caracteres de estas cadenas para almacenarlos

' en dos cuadros de listas y dos arreglos

For I = 1 To 12

   LstMeses.AddItem Trim(Mid(Meses, 9 * (I - 1) + 1, 9))

   Mes(I) = Trim(Mid(Meses, 9 * (I - 1) + 1, 9))

Next

For I = 1 To 7

   LstDias.AddItem Trim(Mid(Dias, 9 * (I - 1) + 1, 9))

   Dia(I) = Trim(Mid(Dias, 9 * (I - 1) + 1, 9))

Next

' Las siguientes líneas extraen Día, Mes y Año de la fecha actual
```

```
Fecha = Date

NDia = Mid(Fecha, 1, 2)

NMes = Mid(Fecha, 4, 2)

NAño = Mid(Fecha, 7, 4)

' La siguiente línea guarda en el cuadro de texto la cadena literal de la fecha

TxtFecha.Text = Trim(Dia(Weekday(Date, 2))) + " " + NDia + " de " + Trim(Mes(NMes)) + " del " + NAño

End Sub
```

Ejemplo 11

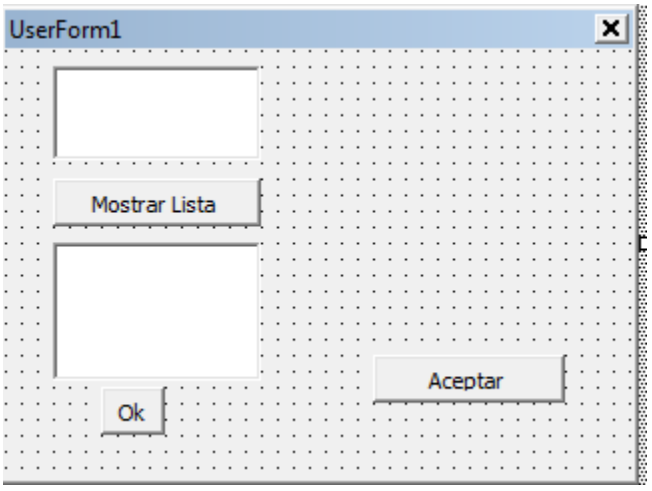
Ahora vamos a generar un formulario que permita colocar los elementos seleccionados en un cuadro de lista

Procedimiento:

Luego de insertar un UserForm, inserte y defina los siguientes cuadros de control:

Control	Nombre, caption
Comando	CmdOk, Ok
Comando	CmdFin, Terminar
Ccomando	CmdListar Mostrar Lista
Cuadro de Lista	LstCaja
Cuadro de texto	TxtCaja

Una muestra del formulario es la siguiente imagen



Primero almacenamos una lista de nombres en el cuadro de lista. Esto lo hacemos en el procedimiento para CmdListar, de manera que, al hacer clic en este botón, se genera la lista..

```
Private Sub CmdListar_Click()

LstCaja.AddItem "Ilmer"

LstCaja.AddItem "Carlos"

LstCaja.AddItem "César"

LstCaja.AddItem "Miguel"

LstCaja.AddItem "Pedro"

LstCaja.AddItem "Bals"

LstCaja.AddItem "Yack"

LstCaja.AddItem "Pipo"

LstCaja.AddItem "Josue"

LstCaja.AddItem "Manolo"

End Sub
```

Luego de seleccionar uno de la lista, al hacer clic en el botón Ok, el elemento seleccionado se visualiza en el cuadro de texto, TxtLista. Esto se aprecia en el procedimiento

```
Private Sub CmdOk_Click()
```

```
' Extrae varios de uno en uno
 TxtCaja.Text = TxtCaja.Text + Chr(13) + LstCaja.List(LstCaja.ListIndex)
' Desactiva lo que está seleccionado
 LstCaja.Selected(LstCaja.ListIndex) = False
' Extrae los que están seleccionados
End Sub
```

La instrucción: LstCaja.Selected(LstCaja.ListIndex) = False

permite desactivar la selección del elemento luego de hacer clic en el botón Ok.

El código completo se muestra a continuación:

```
Private Sub CmdFin_Click()
End
End Sub
Private Sub CmdListar_Click()
LstCaja.AddItem "Ilmer"
LstCaja.AddItem "Carlos"
LstCaja.AddItem "César"
LstCaja.AddItem "Miguel"
LstCaja.AddItem "Pedro"
LstCaja.AddItem "Bals"
LstCaja.AddItem "Yack"
LstCaja.AddItem "Pipo"
LstCaja.AddItem "Josue"
LstCaja.AddItem "Manolo"
```

```
End Sub

Private Sub CmdOk_Click()

' Extrae varios de uno en uno

 TxtCaja.Text = TxtCaja.Text + Chr(13) + LstCaja.List(LstCaja.ListIndex)

' Desactiva lo que está seleccionado

 LstCaja.Selected(LstCaja.ListIndex) = False

' Extrae los que están seleccionados

End Sub

Private Sub LstCaja_Click()

LstCaja.MultiSelect = fmMultiSelectExtended

End Sub
```

Ejemplo 11

El siguiente ejemplo permite listar una columna de datos de una hoja del Excel, seleccionar un elemento y colocarlo en un cuadro de lista.

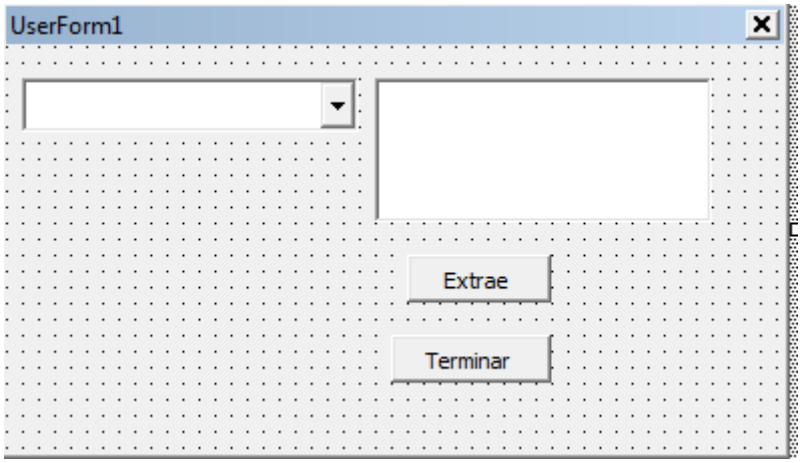
Procedimiento:

Para ello utilizaremos un cuadro combinado para colocar los elementos de la columna de la hoja, un botón de comando para que realice esta transferencia, un cuadro de lista para recibir el elemento seleccionado y un botón de comando para dar por finalizado el formulario. Los datos se encuentran en el libro Formu01.xlsm.

Los botones de control a ser usados se muestra en la siguiente tabla:

Control	Nombre, caption
Comando	CmdListar, Extrae
Comando	CmdFin, Terminar
Cuadro de Lista	LstReporte
Cuadro combinado	CboLista

La imagen siguiente muestra este formulario:



El botón de comando <Terminar> sólo contiene la instrucción End

El botón de comando <Extrae> permite ingresar por teclado el número de fila y columna donde se encuentran los datos y luego clocarlos en el cuadro combinado CboLista.

Esto se realiza con el siguiente procedimiento:

```
Private Sub CmdListar_Click()

Sheets("Tablas").Select

Ir = Val(InputBox("Ingrese la primera fila de la lista"))

Icol = Val(InputBox("Ingrese la columna de la lista"))

Cadena = Trim(Cells(Ir, Icol))

While Len(Cadena) > 0

   CboLista.AddItem Cadena

   Ir = Ir + 1

   Cadena = Trim(Cells(Ir, Icol))

Wend

End Sub
```

Luego de haberse extraído los datos de la hoja, al hacer clic en un elemento de esta lista, el control de lista recibe dicho elemento. Esto se realiza con el siguiente procedimiento:

Observe que los últimos dos procedimientos están por gusto.

```
Private Sub CboLista_Change()

LstReporte.AddItem CboLista.List(CboLista.ListIndex)

End Sub
```

A continuación se muestra todos los procedimientos:

```
Private Sub CboLista_Change()

LstReporte.AddItem CboLista.List(CboLista.ListIndex)

End Sub

Private Sub CmdFin_Click()

End

End Sub

Private Sub CmdListar_Click()

Sheets("Tablas").Select

Ir = Val(InputBox("Ingrese la primera fila de la lista"))

Icol = Val(InputBox("Ingrese la columna de la lista"))

Cadena = Trim(Cells(Ir, Icol))

While Len(Cadena) > 0

    CboLista.AddItem Cadena

    Ir = Ir + 1

    Cadena = Trim(Cells(Ir, Icol))

Wend
```

```
End Sub

Private Sub LstReporte_Click()

End Sub

Private Sub UserForm_Click()

End Sub
```

Nota:

Si la lista fuera más grande se puede modificar la propiedad ListRows del cuadro combinado a fin de que se pueda visualizar más elementos de la lista.

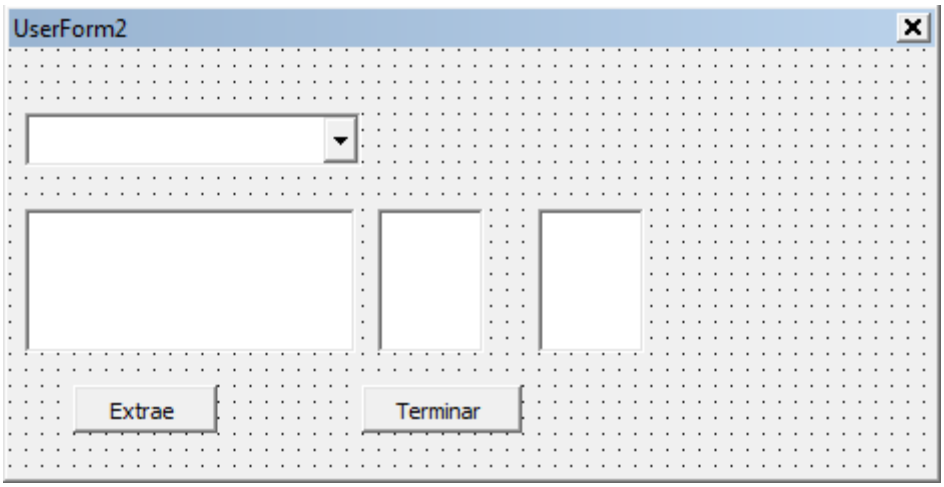
Ejemplo 12

Como en el ejemplo anterior, se desea extraer todos los datos correspondientes a un elemento de una determinada fila de datos colocándolos cada uno de ellos en un control de lista.

Procedimiento

Para ello haremos uso de los mismos controles que el ejemplo anterior con el único cambio en la ubicación de los elementos en el formulario y hemos aumentado dos cuadros de control: LstPrecio y LstCosto.

La imagen del formulario es el siguiente



El código del procedimiento CboLista es el siguiente:

```
Private Sub CboLista_Change()

LstReporte.AddItem CboLista.List(CboLista.ListIndex)
```

```
LstPrecio.AddItem Cells(CboLista.ListIndex + 2, 2)

LstCosto.AddItem Cells(CboLista.ListIndex + 2, 3)

End Sub
```

Reemplace este procedimiento por el anterior y tendrá la solución a este ejemplo.

Ejemplo 13

En este ejemplo se trata de usar un formulario para ingresar datos a través de un panel diseñado en el formulario hacia una hoja de un libro abierto. En particular, supongamos que se trata de ingresar el nombre de un producto, el precio unitario, la cantidad y el monto de un posible descuento. En la hoja debe aparecer una columna en el cual se calcule el monto del IGV (18%) de las ventas. El formulario deberá ser activado desde una macro la cual se iniciará insertando una nueva hoja, dándole un nombre y colocando la cabecera de las columnas de datos.

Procedimiento

Usaremos los siguientes cuadros de control de formulario:

Control	Nombre, caption
Formulario	FrmPanel, Panel de datos
Comando	CmdTransfiere, Transfiere
Comando	CmdFin, Terminar
Cuadro de texto	TxtProducto
Cuadro de texto	TxtPrecio
Cuadro de texto	TxtCantidad
Cuadro de texto	TxtDescuento
Tres Etiquetas	LblEtq1, LblEt2, LblEt3, LblEt4

Este será el módulo correspondiente a la macro:

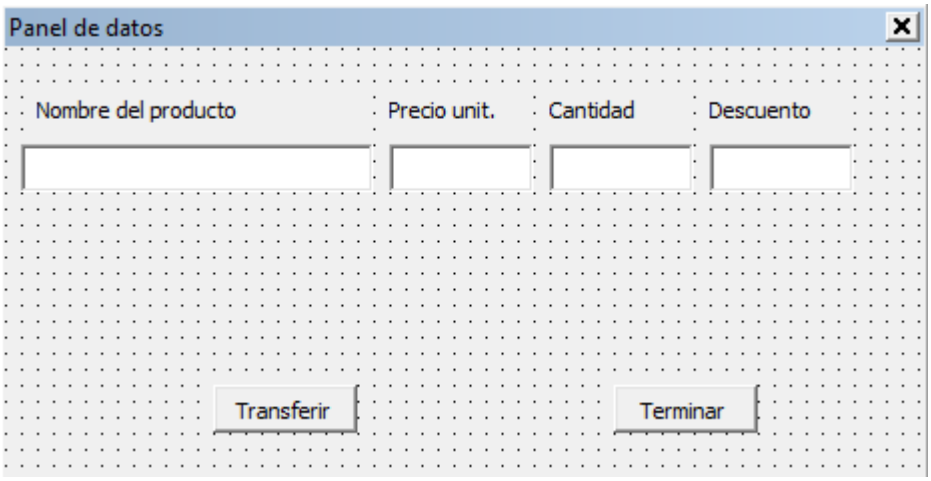
```
Public Sub Inicio()

HojaName = InputBox("Nombre de la hoja a ser creada")
```

```
Sheets.Add
With ActiveSheet
   .Name = HojaName
End With
Range("A2") = "Nombre del producto"
Range("B2") = "Precio unit."
Range("C2") = "Cantidad"
Range("D2") = "Ventas"
Range("E2") = "I.G.V."
Range("F2") = "Descuento"
Range("G2") = "Venta neta"
FrmPanel.Show
End Sub
```

Al iniciarse la ejecución del formulario, el siguiente procedimiento permitirá hacer que el cuadro de texto para el nombre del producto quede activado y se inicialice un contador para las filas (NroDatos).

```
Private Sub UserForm_Click()
TxtProducto.Enabled = True
NroDatos = 0
End Sub
```

La siguiente imagen muestra el panel de datos (formulario)



Para mayor facilidad se puede ingresar los datos usando la tecla <Tab>. En el cuadro de texto para Descuento se debe ingresar 0 si no hay descuento. A continuación se debe hace clic en el botón <Transferir>. El siguiente es el procedimiento que le corresponde a este botón:

```
Private Sub CmdTransferir_Click()

NroDatos = NroDatos + 1

Cells(NroDatos + 2, 1) = TxtProducto.Text

Cells(NroDatos + 2, 2) = TxtPrecio.Text

Cells(NroDatos + 2, 3) = TxtCantidad.Text

Cells(NroDatos + 2, 4) = TxtPrecio.Value * TxtCantidad.Value

Cells(NroDatos + 2, 5) = Cells(NroDatos + 2, 4) * 0.18

Cells(NroDatos + 2, 6) = TxtDescuento.Text

Cells(NroDatos + 2, 7) = Cells(NroDatos + 2, 4) + Cells(NroDatos + 2, 5) - TxtDescuento.Value

TxtProducto.Text = ""

TxtPrecio.Text = ""

TxtCantidad.Text = ""

TxtDescuento.Text = ""

TxtProducto.SetFocus
```

End Sub

Observe que luego de ingresar los datos y los valores calculados a celdas de la nueva hoja, los controles son puestos en blanco y con la última instrucción se vuelve activar el cuadro de texto Producto que, de otra manera, se tendría que hacer clic en dicho cuadro.

A continuación se muestra todos los procedimientos del formulario.

```
Dim NroDatos As Integer

Private Sub CmdFin_Click()

End

End Sub

Private Sub CmdTransferir_Click()

NroDatos = NroDatos + 1

Cells(NroDatos + 2, 1) = TxtProducto.Text

Cells(NroDatos + 2, 2) = TxtPrecio.Text

Cells(NroDatos + 2, 3) = TxtCantidad.Text

Cells(NroDatos + 2, 4) = TxtPrecio.Value * TxtCantidad.Value

Cells(NroDatos + 2, 5) = Cells(NroDatos + 2, 4) * 0.18

Cells(NroDatos + 2, 6) = TxtDescuento.Text

Cells(NroDatos + 2, 7) = Cells(NroDatos + 2, 4) + Cells(NroDatos + 2, 5) - TxtDescuento.Value

TxtProducto.Text = ""

TxtPrecio.Text = ""

TxtCantidad.Text = ""

TxtDescuento.Text = ""

TxtProducto.SetFocus
```

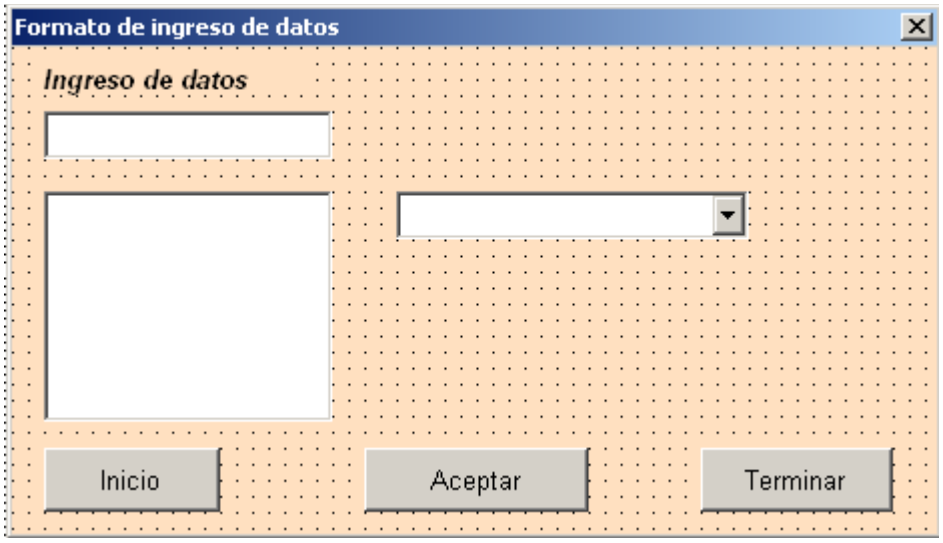
```
End Sub

Private Sub UserForm_Click()

TxtProducto.Enabled = True

NroDatos = 0

End Sub
```

Ejemplo 14

El siguiente ejemplo permite manipular datos provenientes desde un cuadro de texto, ingresados por teclado y los seleccionados de una lista, almacenada en un cuadro combinado.

Procedimiento

A continuación se muestra una imagen del formulario.



Al diseñarlo debe insertar un cuadro de texto cuyo nombre sea TxtDatos; un control de lista, cuyo nombre sea LstLista01, un cuadro combinado que tenga por nombre CboInfo01 y tres botones de comando: CmdInicio, CmdAceptar y CmdFin; sus captions se muestra en la imagen.

El código es el siguiente:

```
Private Sub CmdAceptar_Click()

LstLista01.AddItem TxtDatos.Value

LstLista01.AddItem CboInfo01.List(CboInfo01.ListIndex)
```

```
TxtDatos.Text = ""

TxtDatos.SetFocus

End Sub

Private Sub CmdFin_Click()

n = LstLista01.ListCount

For i = 1 To n

Cells(i, 3) = LstLista01.List(i - 1)

Next

End

End Sub

Private Sub CmdInicio_Click()

TxtDatos.Enabled = True

LstLista01.Clear

CboInfo01.Clear

TxtDatos.Text = ""

Sheets(1).Activate

n = InputBox("Nro. de filas")

Cells(1, 1).Select

For i = 1 To n

    CboInfo01.AddItem Cells(i, 1)

Next
```

Nota:

Observe que no siempre el botón Terminar debe tener una única instrucción. El archivo se llama **Formulario01.xlsm**.

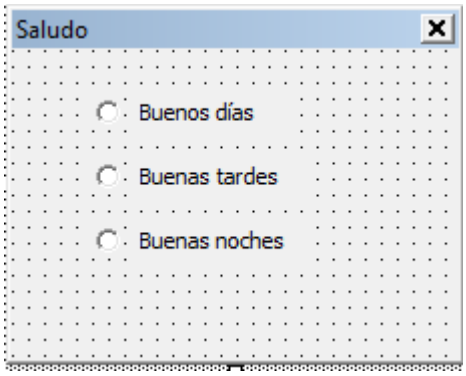
1.3. Botón de opción

Este control permite seleccionar un elemento y sólo uno, de una lista de opciones. El procedimiento que le corresponde al hacer clic es:

```
Private Sub OptionButton1_Click()

End Sub
```

Ejemplo 15

En este ejemplo ingresaremos tres botones de opción. Al hacer clic en cada uno de ellos, se debe emitir un mensaje. La siguiente imagen es una muestra de este formulario:



Este es el código para este formulario:

```
Private Sub OptDias_Click()

MsgBox "Hola amigo, Buenos días ..."

End Sub

Private Sub OptNoches_Click()

MsgBox "Hola amigo, Buenas noches"

End Sub

Private Sub Opttardes_Click()

MsgBox "Hola amigo, Buenas tardes ..."

End Sub
```

Nota:

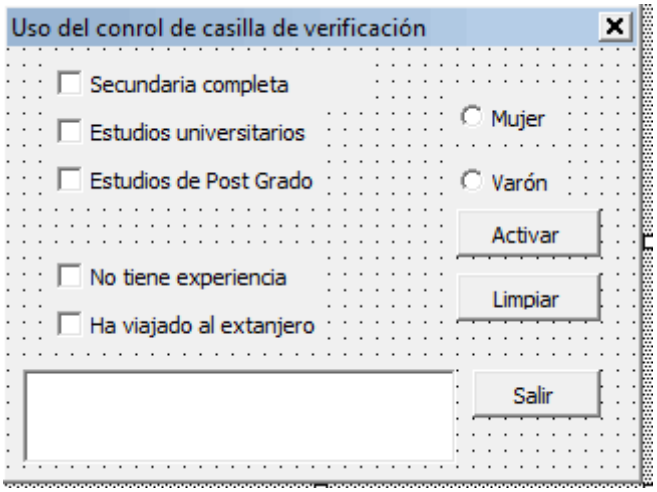
Sugerimos a nuestro amigo lector que inserte un botón de comando a fin de tener una forma de terminar el uso de este formulario. Como lo hemos hecho con todos los formularios colocando ***End*** al interior del código del botón.

Control Casilla de Verificación

Este control permite seleccionar uno o más elementos de una lista de elementos. A diferencia de control anterior en el que sólo se selecciona uno, en este puede seleccionarse más de uno a la vez.

Ejemplo 16

En este ejemplo colocaremos un grupo de controles de verificación y dos botones de opción a fin de seleccionar apropiadamente lo que corresponda. El resultado de la selección se recibirá en un cuadro de texto, que tendrá activada la propiedad multilínea. La imagen del formulario es la siguiente:



Los nombres de cada control así como su Caption se da en la siguiente tabla.

Control	Nombre, caption
Formulario	FrmCasilla, Uso del control de verificación
Opción	Opt00, Mujer
Opción	Opt01, Varón
Casilla de verificación	Chk00, Secundaria completa
Casilla de verificación	Chk01, Estudios universitarios
Casilla de verificación	Chk02, Estudios de Post Grado

Casilla de verificación	Chk03, No tiene experiencia
Casilla de verificación	Chk04, Ha viajado al extranjero
Comando	CmdActivar, Activar
Comando	CmdClear, Limpiar
Comando	CmdFin, Salir
Cuadro de texto	TxtLista, Propiedad: Mulitiline: True

El código completo se muestra en las siguientes líneas

```
Private Sub CmdActivar_Click()

If Opt00 Then

    TxtLista.Text = "Mujer con "

Else

    TxtLista.Text = "Hombre con "

End If

If Chk00 Then TxtLista.Text = TxtLista.Text + Chk00.Caption + Chr(13)

If Chk01 Then TxtLista.Text = TxtLista.Text + Chk01.Caption + Chr(13)

If Chk02 Then TxtLista.Text = TxtLista.Text + Chk02.Caption + Chr(13)

If Chk03 Then TxtLista.Text = TxtLista.Text + Chk03.Caption + Chr(13)

If Chk04 Then TxtLista.Text = TxtLista.Text + Chk04.Caption + Chr(13)

End Sub

Private Sub CmdClear_Click()

Chk00.Value = False

Chk01.Value = False

Chk02.Value = False
```

```
Chk03.Value = False

Chk04.Value = False

Opt00.Value = False

Opt01.Value = False

TxtLista.Text = ""

End Sub

Private Sub CmdSalir_Click()

End

End Sub
```

Nota:

El libro Paneles.xlsm contiene los últimos formularios de estos ejemplos.

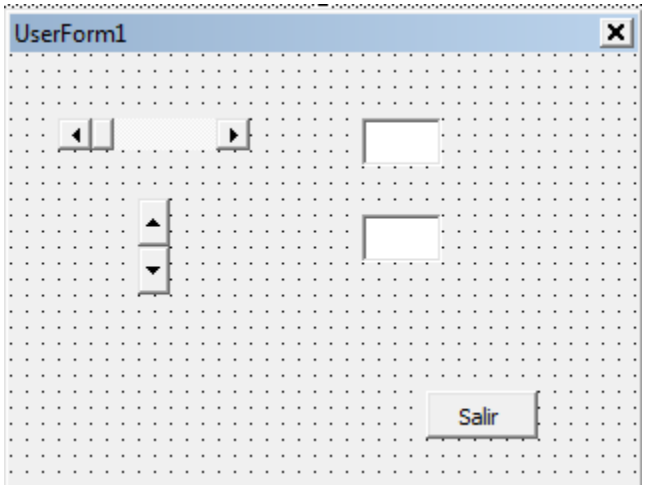
Control de Barra de desplazamiento y Control Botón de número

Estos controles permiten asignar un valor a una variable o a otro objeto, para un valor del desplazamiento que se haya realizado en dicho botón.

Ejemplo 17

Este ejemplo permite seleccionar un valor para el peso y otro para la edad, los cuales se visualizan en sus respectivos cuadros de texto.

La siguiente imagen muestra dicho formulario



La siguiente tabla muestra los nombres de los elementos de este formulario

Control	Nombre, caption
Formulario	FrmBarra
Barra de desplazamiento	Scpeso
Número	NroEdad
Cuadro de texto	TxtPeso
Cuadro de texto	TxtEdad
Comando	CmdFin, Salir

El código correspondiente se muestra en el siguiente listado

```
Private Sub CmdFin_Click()

End

End Sub

Private Sub NroEdad_Change()

TxtEdad.Text = NroEdad.Value

End Sub

Private Sub ScPeso_Change()

TxtPeso.Text = ScPeso.Value

End Sub
```

Unidad 7. Aplicaciones

A continuación vamos a desarrollar dos aplicaciones del uso macros y formularios en la solución de ciertos problemas

Aplicación 1 : Consulta y extracción en una base de datos

Se desea contar con una macro que permita consultar si es cliente o no y si existe, extraer toda la información de uno o más clientes hacia una nueva hoja. La macro debe solicitar el nombre de la hoja hacia donde se desea extraer y también el DNI para realizar la búsqueda. Los datos se encuentran en el archivo **QbdClientes.xlsm**.

Procedimiento:

P1. Insertamos un módulo en el editor. Digitamos: Sub Consulta y presionamos <Enter> para crear el módulo:

```
Sub Consulta()

End
```

P2. Declaramos las variables HjName y Dni como de tipo String. El primero para recibir el nombre de la hoja y el segundo para recibir el número de DNI que se consultará.

P3. El siguiente código permite saber el número de registro que contiene la base de datos (menos las dos filas iniciales).

```
nDat = Columns("A:A").Range("A65536").End(xlUp).Row
```

P4. El siguiente segmento de código permite realizar la búsqueda en toda la tabla por el DNI solicitado, usando a Ix como puntero de fila. Si lo encuentra Ix contendrá el número de fila desde donde se extraerán los datos.

```
For Ix = 3 To nDat

    If Val(Trim(Cells(Ix, 1))) = Dni Then

        Sheets(HjName).Cells(Iz, 1) = Dni

        Sheets(HjName).Cells(Iz, 2) = Cells(Ix, 2)

        Sheets(HjName).Cells(Iz, 3) = Cells(Ix, 3)
```

```
            Sheets(HjName).Cells(Iz, 4) = Cells(Ix, 4)
            Sheets(HjName).Cells(Iz, 5) = Cells(Ix, 5)
            Sheets(HjName).Cells(Iz, 6) = Cells(Ix, 6)
            Sheets(HjName).Cells(Iz, 7) = Cells(Ix, 7)
            Sheets(HjName).Cells(Iz, 8) = Cells(Ix, 8)
            Ir = 1
        End If
    Next
```

P5. El siguiente segmento emite un mensaje si no encuentra el DNI

```
    If Ir = 0 Then
        MsgBox ("No existe este DNI(RUC). Verifique")
        If Iz > 2 Then Iz = Iz - 1
    End If
```

P6. Finalmente el siguiente código permite retornar a la hoja de datos y saber si se desea hacer otra búsqueda.

```
    Sheets(HjName).Activate
    Range("A5").Select
    If UCase(InputBox("Desea extraer otro? (S/N)")) = "S" Then
        Iz = Iz + 1
    Else
        Iz = 0
    End If
```

El procedimiento completo se encuentra en el editor del Visual basic. Hemos insertado un botón de comando a fin de ejecutar la macro haciendo clic en él. Se puede hacer clic en C1 de la hoja Salida para volver a la hoja de datos.

Aplicación 2. Emisión de un reporte de compra.

Se desea desea obtener un documento que contenga la información de cierto cliente. Para ello se cuenta con dos bases de datos: Productos y Clientes. Se debe diseñar un formato en una hoja del Excel y en ella debe seleccionarse el código de un cliente desde un cuadro combinado. Después de visualizar el formato con los datos se debe transferir a una hoja de datos Resumen.

Procedimiento

Como se puede apreciar, al seleccionar el código del cliente, se debe obtener toda la información. En todos los casos es una simple búsqueda en tablas. El ítem que se busca debe ser el código del producto y también se debe buscar por el nombre del cliente. Para ello es suficiente insertar en dond corresponda dos cuadros combinados. Luego usar la función Indice en todos los casos.

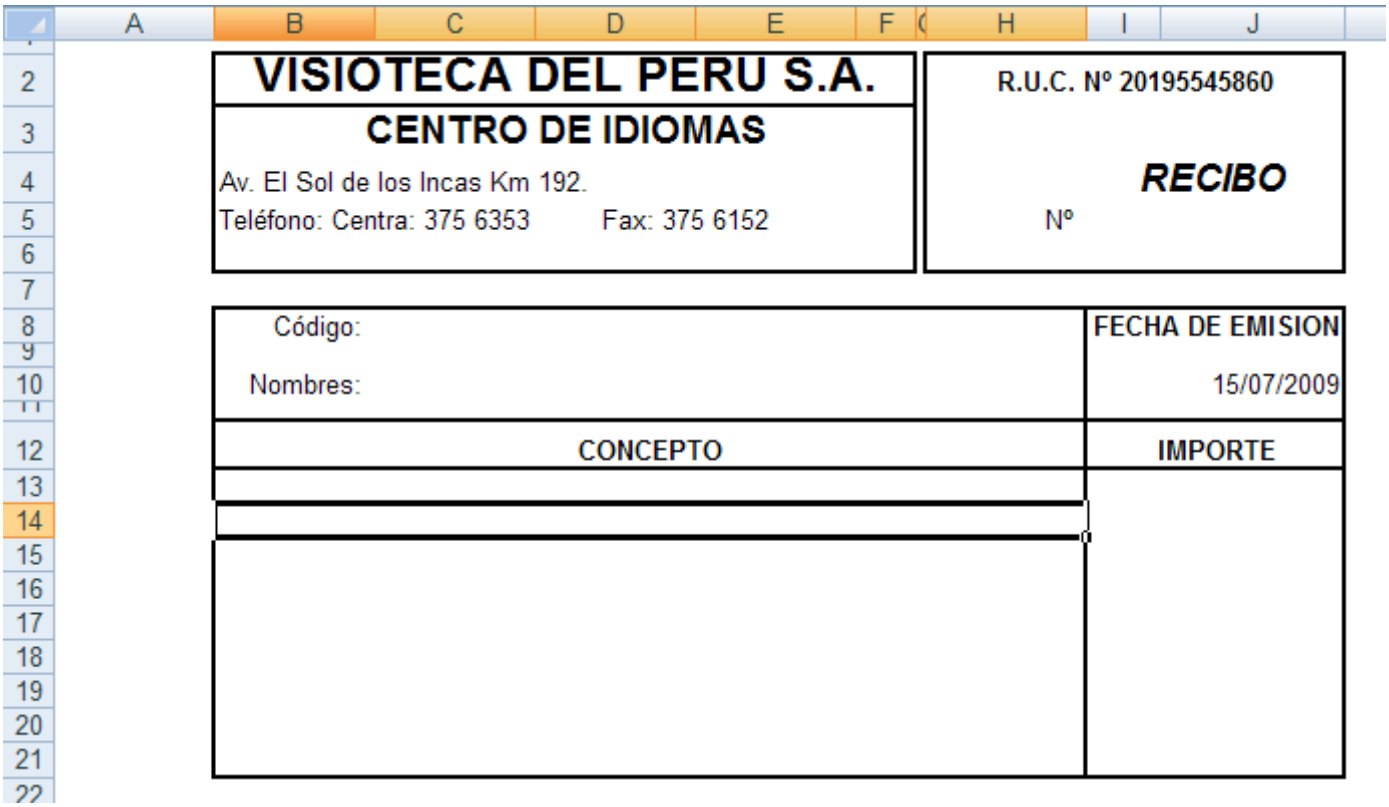
La solución se muestra en el archivo **Detalle de compra.xls**.

Aplicación 3. Macro para imprimir un formato-recibo.

En el siguiente problema se dispone de un formato de recibo que se genera ingresando los datos de un cliente. Luego de completar todos los datos, se debe hacer uso de una macro que permita imprimir el rango seleccionado.

Procedimiento:

P1. Diseñar el siguiente formato de recibo:



P2. Suponiendo que ya se han ingresado los datos, se pasará a seleccionar el rango que incluye el formato.

P3. Se grabará una macro que permita imprimir todo lo seleccionado. La macro se llama MacPrint. Hemos colocado un botón de comando al cual le hemos asignado dicha macro. La macro se encuentra en el archivo **Formato recibo.xlsm**.

Aplicación 4. Realizar varias consultas y transferir al Excel.

El siguiente caso permite extraer datos de una hoja de trabajo y los devuelve hacia otra hoja.

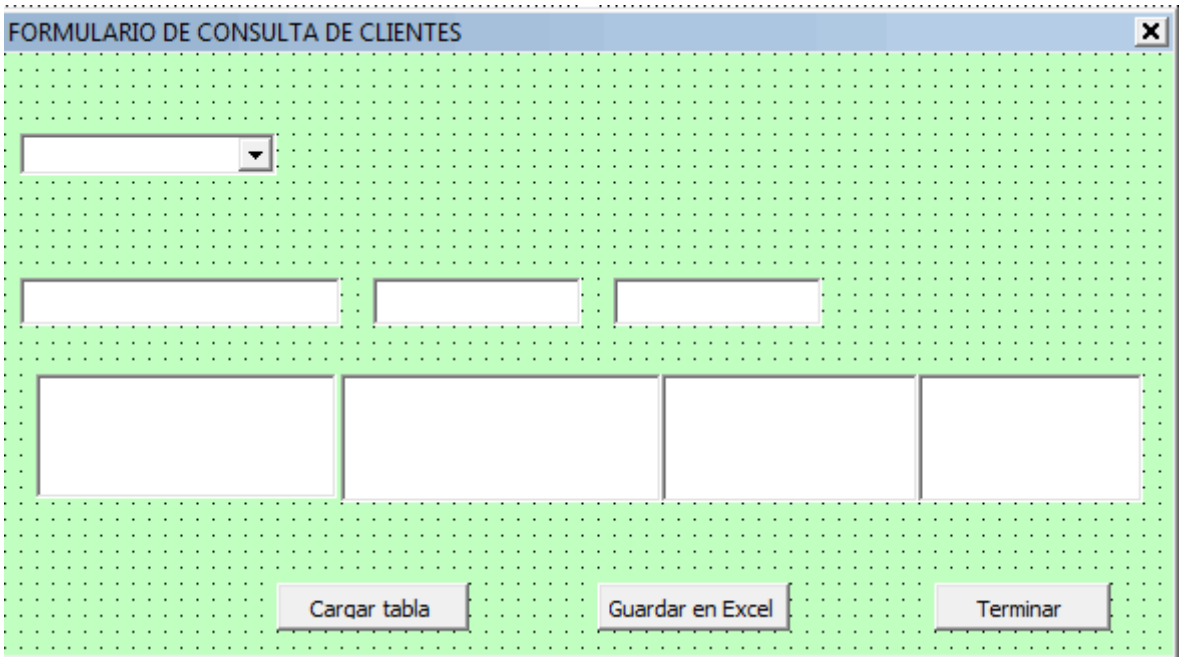
La siguiente imagen es una muestra del formulario que se desea crear.

Los cuadros de control a ser usados son:

Un cuadro combinado. Para mostrar la lista de los clientes y seleccionar uno.

Tres cuadros de texto. Para ver los datos que se desean almacenar

Tres cuadros de lista. Para almacenar los datos seleccionados



Los procedimientos que resuelven este problema se muestra a continuación

```
Dim NDat As Integer

Private Sub CboDatos_Change()

TxtDirec.Text = Cells(CboDatos.ListIndex + 2, 2)

TxtRuc.Text = Cells(CboDatos.ListIndex + 2, 3)

TxtTelef.Text = Cells(CboDatos.ListIndex + 2, 4)

LstNombre.AddItem CboDatos.List(CboDatos.ListIndex)

LstDirec.AddItem Cells(CboDatos.ListIndex + 2, 2)

LstRuc.AddItem Cells(CboDatos.ListIndex + 2, 3)

LstTelef.AddItem Cells(CboDatos.ListIndex + 2, 4)

NDat = NDat + 1
```

```
End Sub

Private Sub CmdFin_Click()

End

End Sub

Private Sub CmdInit_Click()

Sheets("Clientes").Activate

Rango = "Cliente"

NRow = Range(Rango).Count

For I = 0 To NRow - 1

   CboDatos.AddItem Cells(I + 2, 1)

Next

NDat = 0

End Sub

Private Sub Cmdstore_Click()

For I = 1 To NDat

   Sheets("Lista").Cells(I + 1, 1) = LstNombre.List(I - 1)

   Sheets("Lista").Cells(I + 1, 2) = LstDirec.List(I - 1)

   Sheets("Lista").Cells(I + 1, 3) = LstRuc.List(I - 1)

   Sheets("Lista").Cells(I + 1, 4) = LstTelef.List(I - 1)

Next

End Sub
```

Para ver la solución al problema, abra el archivo **Consulta con formulario.xlsm**.

Aplicación 5. Seleccionar determinado producto para su venta y transferir al Excel

La siguiente imagen contiene el formulario que usaremos para resolver este problema.

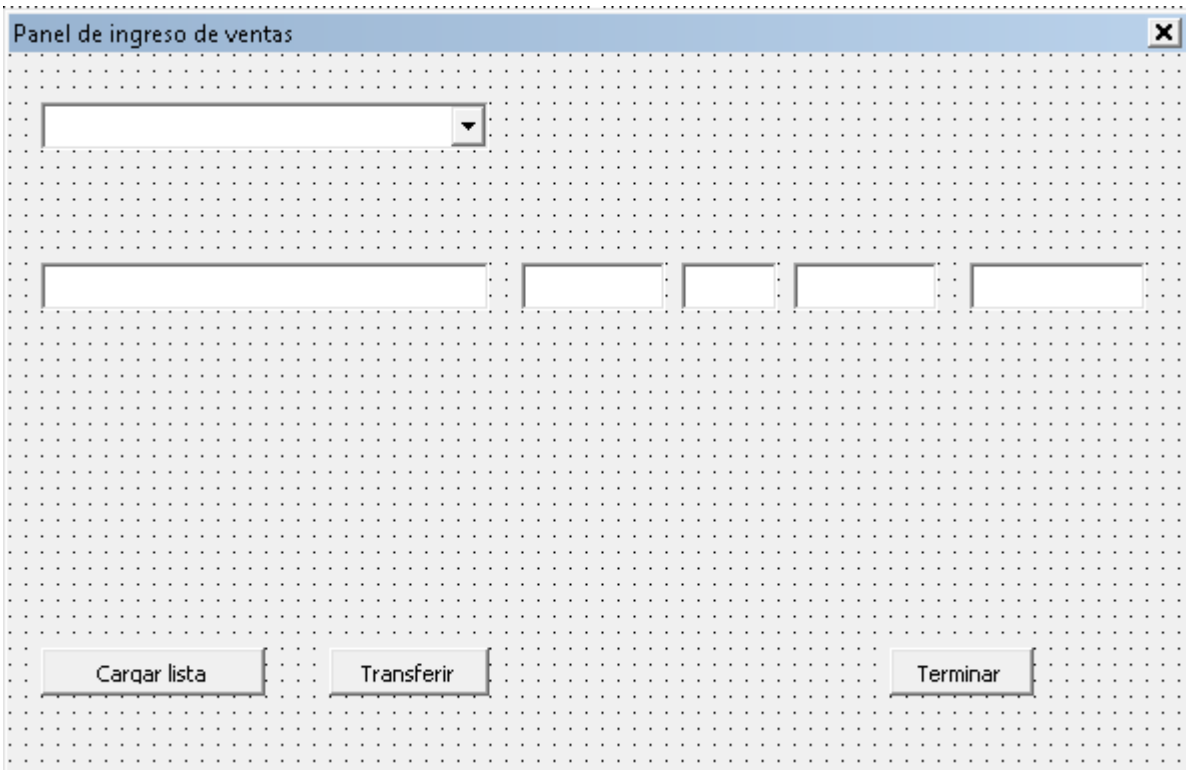
En ella se ha usado:

Un cuadro combinado: CboProductos: Para contener la lista de los productos

Cinco cuadros de texto: TxtProducto, TxtPrecio, TxtCantidad, TxtMonto y TxtFecha.

El primero contiene el elemento seleccionado y el precio que le corresponde. Se ingresa la cantidad a vender y en TxtMonto se obtiene el producto de precio por cantidad. El cuadro de texto TxtFecha contiene la fecha del día. El formulario se llama FrmVentas.

Para activar este formulario desde una hoja del Excel, se usa un botón de comando el cual está asignado a una macro contenida en el módulo 1 que contiene el procedimiento IngresoVentas.



A continuación presentamos el código del procedimiento y el código de cada uno de los elementos del formulario.

```
Sub IngresoVentas()

Workbooks("Ejemplo 3.xlsm").Activate
```

```
Hoja = Trim(InputBox("Nombre de hoja. Si va a crear, presione <Intro>"))
If Len(Hoja) > 0 Then
   Sheets(Hoja).Activate
Else
   Sheets.Add
   ActiveSheet.Name = "Tempo"
   Cells(1, 26) = 1
   Cells(1, 1) = "Producto"
   Cells(1, 2) = "Precio"
   Cells(1, 3) = "Cantidad"
   Cells(1, 4) = "Monto"
   Cells(1, 5) = "Fecha"
End If
FrmVentas.Show
```

```
End Sub
```

```
Dim Hoja As Variant
Private Sub CboProductos_Change()
TxtProducto.Text = CboProductos.List(CboProductos.ListIndex)
TxtPrecio.Text = Sheets("Productos").Cells(CboProductos.ListIndex + 2, 2)
End Sub
Private Sub CmdFin_Click()
End
End Sub
```

```
Private Sub CmdLoad_Click()
Hoja = ActiveSheet.Name
Sheets("Productos").Activate
iFila = 2
Cadena = Trim(Cells(iFila, 1))
While Len(Trim(Cadena)) > 0
    CboProductos.AddItem Cadena
    iFila = iFila + 1
    Cadena = Trim(Cells(iFila, 1))
Wend
End Sub
Private Sub CmdTransf_Click()
Sheets(Hoja).Select
Ix = Cells(1, 26)
Ix = Ix + 1
Cells(Ix, 1) = TxtProducto.Text
Cells(Ix, 2) = TxtPrecio.Text
Cells(Ix, 3) = TxtCantidad.Text
Cells(Ix, 4) = TxtMonto.Text
Cells(Ix, 5) = TxtFecha.Text
Cells(1, 26) = Ix
TxtProducto.Text = ""
TxtPrecio.Text = ""
```

```
TxtCantidad.Text = ""

TxtMonto.Text = ""

TxtFecha.Text = ""

End Sub

Private Sub TxtCantidad_Change()

TxtMonto.Text = Val(TxtCantidad.Text) * Val(TxtPrecio.Text)

TxtFecha.Text = Date

End Sub

Private Sub TxtPrecio_Change()

TxtCantidad.SetFocus
```

```
End Sub
```