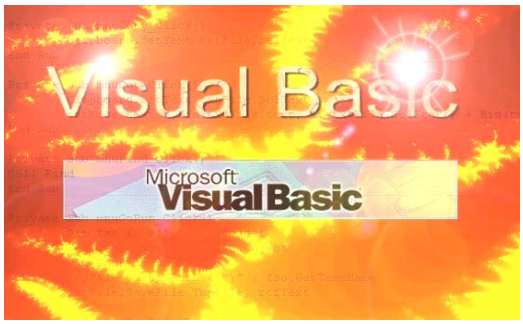
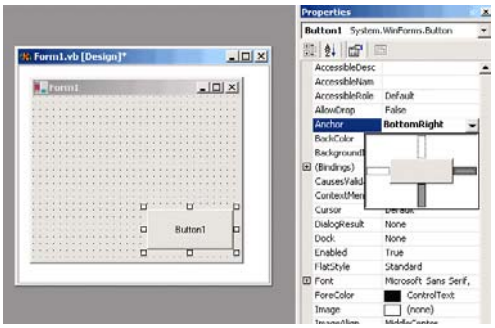
CURSO

Curso Completo de Visual Basic 6.0





Escuela Superior de Ingenieros Industriales
UNIVERSIDAD DE NAVARRA
Javier García de Jalón · José Ignacio Rodríguez
Alfonso Brazález · Patxi Funes · Eduardo Carrasco · Jesús Calleja

4.4 CAJAS DE DIÁLOGO ESTÁNDAR (CONTROLES COMMON DIALOG)

El control de ***cuadro de diálogo estándar*** de ***Windows 95/98SE/XP..*** (***Common Dialog***) ofrece una forma sencilla y eficiente de realizar algunas de las tareas más comunes de un programa, tales como la selección de un fichero para lectura/escritura, la impresión de un fichero o la selección de un tipo de letra o un color.

Lo primero que hay que hacer es ubicar el control en el formulario. El control se representará como un icono de tamaño invariable. No es posible especificar la ubicación que tendrá la caja de diálogo en la pantalla, ya que se trata de una propiedad no accesible por el usuario.

Un único ***cuadro de diálogo estándar*** puede bastar para realizar todas las funciones que se deseen, es decir, no es necesario insertar un cuadro de diálogo para imprimir un texto y otro para guardarlo, sino que ambos pueden compartir el mismo cuadro de diálogo simplemente invocando a uno u otro tipo en tiempo de ejecución (no es posible indicarlo en tiempo de diseño). Para ello se dispone de los métodos siguientes***: ShowColor, ShowFont, ShowHelp, ShowOpen, ShowPrinter*** y ***ShowSave.*** En ocasiones interesará introducir varios controles diferentes por motivos de claridad o para que ciertas propiedades sean distintas.

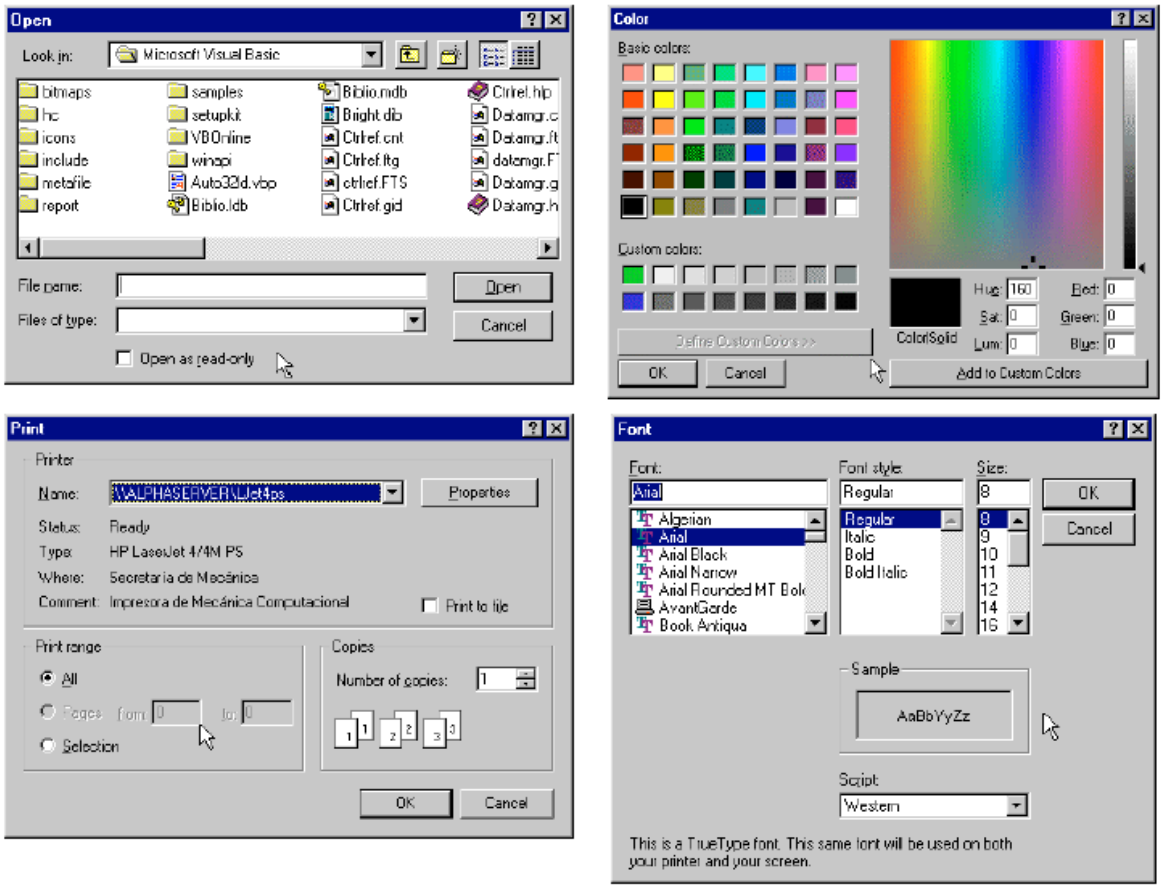


Figura 4.4. Controles Common Dialog.

En la **Figura 4.4** se pueden observar los distintos tipos de control ***Common Dialog***. Por ejemplo, si se desea visualizar un cuadro de diálogo para abrir un fichero, habrá que escribir:

```
dlgAbrir.ShowOpen
```

donde ***dlgAbrir*** es el nombre asignado al control ***Common Dialog***.

Las principales ***propiedades*** de este control en cada una de sus variantes se explican en los apartados siguientes. La propiedad ***Flag*** existe para todos los controles y determina algunas de sus propiedades más importantes.

4.4.1 Open/Save Dialog Control

Las propiedades más importantes de este control son:

- ***DefaultExt***: Es la extensión por defecto a utilizar para abrir/salvar archivos. Con ***Save***, si el nombre del fichero se teclea sin extensión, se añade esta extensión por defecto.

- ***DialogTitle***: Devuelve o da valor al título de la caja de diálogo (cadena de caracteres).

- ***FileName***: Nombre completo del archivo a abrir/salvar, incluyendo el *path*.

- ***FileTitle***: Nombre del archivo a abrir/salvar pero sin la ruta de acceso correspondiente.

- ***Filter***: Contiene los filtros de selección que aparecerán indicados en la parte inferior de la pantalla en la lista de tipos de archivo. Pueden indicarse múltiples tipos de archivo, separándolos mediante un barra vertical (***Alt Gr +< 1>***). Su sintaxis es la siguiente:

```
Objeto.Filter = "(descripción a aparecer en la listbox)|filtro"
```

Por ejemplo:

```
"Texto (*.txt)|*.txt|Imágenes(*.bmp;*.ico)|*.bmp;*.ico"
```

- ***FilterIndex***: Indica el índice (con respecto a la lista de tipos) del filtro por defecto. Se empiezan a numerar por "1".

- ***InitDir***: Contiene el nombre del directorio por defecto. Si no se especifica, se utiliza el directorio actual.

- ***Flags***: Esta propiedad puede tomar muchos valores con objeto de controlar los detalles concretos de este control (por ejemplo, abrir un fichero en modo read only, avisar antes de escribir sobre un fichero ya existente, etc.). Estos valores están definidos por constantes de ***Visual Basic 6.0*** cuyos nombres empiezan con las letras ***cdl***. Para más información en el ***Help*** de ***Common Dialog Control*** buscar ***Properties, Flags Properties (Open, Save As Dialogs)***. Por ejemplo, el valor definido por la constante ***cdlOFNOverwritePrompt*** hace que antes de escribir en un fichero ya existente se pida confirmación al usuario.
  Para establecer varias opciones a la vez se le asigna a ***Flags*** la suma de las constantes correspondientes. Las distintas constantes disponibles se pueden encontrar en el ***Help*** buscando ***Constants/CommonDialog Control***.

4.4.2 Print Dialog Control

Las propiedades más importantes de este control son:

- ***Copies***: Determina el número de copias a realizar por la impresora.

- ***FromPage***: Selecciona el número de página a partir del cual comienza el rango de impresión.

- ***ToPage***: Selecciona el número de página hasta la cual llega el rango de impresión.

- ***PrinterDefault***: Cuando es ***True*** se imprime en el objeto ***Visual Basic Printer***. Además las opciones actuales de impresión que se cambien serán asignadas como las opciones de impresión por defecto del sistema.

- ***Flags***: Ver con ayuda del ***Help*** los posibles valores de esta propiedad.

4.4.3 Font Dialog Control

Las propiedades más importantes de este control son:

- ***Color***: Color de impresión. Para usar esta propiedad hace falta establecer la propiedad ***Flags*** al valor de la constante ***cdlCFEffects***.

- ***FontBold, FontItalic, FontStrikethru, FontUnderline***: Devuelve o asigna los valores de los estilos de la fuente actual.

- ***FontName:*** Devuelve o asigna el nombre de la fuente en uso.

- ***FontSize:*** Devuelve o asigna el tamaño de la fuente en uso.

- ***Min y Max:*** Asigna o lee los valores del tamaño de fuente mínimo y máximo respectivamente que aparecerán en la lista de selección de tamaños de la fuente.

- ***Flags***: Indica si los tipos de letra que se van a mostrar son los de la pantalla (***cdlCFScreenFonts***), los de la impresora (***cdlCFPrinterFonts***) o ambos (***dlCFBoth***). Con la constante ***cdlCFEffects*** se puede indicar que se permite cambiar efectos como el color, subrayado y cruzado con una línea. Si ***Flags*** vale 0 da un error en tiempo de ejecución inndicando que no hay ***fonts*** instaladas.

4.4.4 Color Dialog Control

Las propiedades más importantes de este control son:

- ***Color***: Devuelve o asigna el valor del color actual.

- ***Flags***: Ver con ayuda del ***Help*** los posibles valores de esta propiedad. Por ejemplo, con el valor ***cdlCCFullOpen*** muestra el cuadro de diálogo completo, mientras que el valor ***cdlCCPreventFullOpen*** muestra sólo los colores predefinidos, impidiendo definir otros nuevos. Con el valor ***cdlCCRGBInit*** se establece el color inicial para el cuadro de diálogo.

4.5 FORMULARIOS MÚLTIPLES

Un programa puede contener más de un formulario. De hecho, habitualmente los programas contienen múltiples formularios. Recuérdese que el formulario es la ventana de máximo nivel en la que aparecen los distintos controles.

Sin embargo, un programa siempre debe tener un ***formulario principal***, que es el que aparece al arrancar el programa. Se puede indicar cuál debe ser el formulario principal en el menú ***Project/Project Properties***, en la lengüeta ***General***, en la sección ***Startup Form***.

Por defecto, el programa considera como formulario principal el primero que se haya creado. El resto de formularios que se incluyan en el programa serán cargados en su momento, a lo largo de la ejecución del programa.

En tiempo de diseño para añadir nuevos formularios al programa, hay que acudir al menú ***Project/Add Form***. La forma de cargar y descargar estos formularios se ha explicado con anterioridad. Es importante sin embargo recordar que conviene descargar aquellos sub-formularios que ya no sean de utilidad, ya que así se ahorran recursos al sistema.

Para activar en tiempo de ejecución un formulario distinto del inicial (o del que esté activo en ese momento), se utiliza el método ***Show*** (***frmName.Show***). El método ***Hide*** oculta el formulario, pero lo deja cargado; el método ***Activate*** lo vuelve a mostrar. El método ***Unload*** elimina los elementos gráficos del formulario, pero no las variables y el código. El método ***Unload Me*** descarga el propio formulario que lo llama. Para eliminar completamente un formulario se puede utilizar el comando:

```
Set frmName = NOTHING
```

que llama al evento ***Terminate*** (hay que utilizar también los métodos ***Hide*** o ***Unload*** para que desaparezca de la pantalla).

Para referirse desde un formulario a los objetos y variables de otro formulario se utiliza el ***operador punto*** (*frmName.Object.Property*).
En ciertas ocasiones se desea que el programa no realice ninguna acción hasta que el usuario cierre una ventana o formulario en la que se le pregunta algo o en la que tiene que tomar alguna decisión. En esos casos, al utilizar el método ***Show***, es necesario utilizar el argumento ***style*** con valor 1. A esto se le llama mostrar una ventana en forma ***modal***. Esto quiere decir que no se permitirá al usuario hacer activa ninguna pantalla hasta que el usuario cierre esa ventana modal.

Esto se hace así:

```
frmName.Show 1
```

o bien,

```
frmName.Show vbModal
```

4.5.1 Formularios y sub-formularios

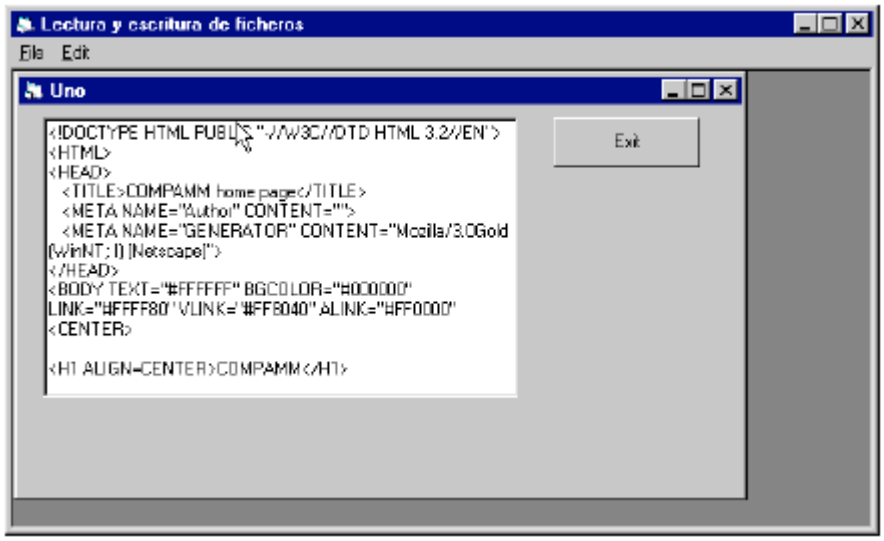


Figura 4.5. Formularios MDI (Multiple Document Interface).

En algunos casos puede ser interesante establecer una jerarquía entre las ventanas o formularios que van apareciendo sucesivamente en la pantalla del ordenador, de tal manera que al cerrar una que se haya establecido como principal, se cierren también todas las que se han abierto desde ella y dentro de ella. De esta forma una misma aplicación puede tener varios documentos abiertos, uno en cada ventana hija. Así trabajan por ejemplo ***Word*** y ***Excel***, que pueden tener varios documentos abiertos dentro de la ventana principal de la aplicación. En el mundo de las ***Windows*** de ***Microsoft*** a esto se llama ***MDI*** (***Multiple Document Interface***). La Figura 4.5 muestra un ejemplo de formulario MDI.

En ***Visual Basic 6.0*** estos formularios que tienen sub-formularios hijos se conocen como ***MDIForms***. Los formularios ***MDI*** se crean desde el menú de ***Visual Basic 6.0*** con el comando ***Project/Add MDI Form***. En una aplicación sólo puede haber un formulario ***MDI***, pero éste puede tener varios hijos. Si se quiere que un formulario sea ***Child***, debe tener su propiedad ***MDIChild*** como ***True***.

Si al iniciar una aplicación el formulario que se carga en primer lugar es un formulario ***Child***, el formulario ***MDI*** se carga al mismo tiempo. Al cerrar un formulario ***MDIForm*** se cierran todossus formularios ***Child***; por ejemplo, al cerrar ***Word*** también se cierran todos los documentos que estuvieran abiertos. Los formularios ***Child*** se minimizan y maximizan dentro de los límites del formulario ***MDI***. Cuando están maximizados, su ***Caption*** aparece junto al ***Caption*** del formulario ***MDI***. Los formularios ***Child*** no tienen menús propios, sino que sus menús aparecen en la barra de menús del formulario ***MDI***.

En una aplicación con un formulario ***MDI*** y uno o más formularios ***Child***, puede haber otros formularios que no sean ***Child*** y que se abren fuera de los límites del formulario ***MDI*** cuando son requeridos.

4.6 ARRAYS DE CONTROLES

Un array de controles esta formado por controles del mismo tipo que comparten el nombre y los procedimientos o funciones para gestionar los eventos. Para identificar a cada uno de los controles pertenecientes al array se utiliza ***Index*** o *índice*, que es una propiedad más de cada control.
Suponiendo que el sistema tenga memoria suficiente un array en ***Windows*** podría llegar a tener hasta 32767 elementos.
La utilidad principal de los arrays se presenta en aquellos casos en los que el programa debe responder de forma semejante a un mismo evento sobre varios controles del mismo tipo. Los ejemplos más claros son los botones de opción y los menús. En estos casos el programa responde de manera semejante independientemente de cuál es la opción seleccionada.

Los arrays de controles comparten código, lo cual quiere decir que sólo hay que programar una función para responder a un evento de un determinado tipo sobre cualquier control del array. Las funciones que gestionan los eventos de un array tienen siempre u argumento adicional del tipo ***Index As Integer*** que indica qué control del array ha recibido el evento.
Una opción avanzada de ***Visual Basic 6.0*** permite crear objetos en tiempo de ejecución, siempre que sean elementos de un array ya existente, con la instrucción ***Load***. De forma análoga se pueden destruir con ***Unload***.

5. MENÚS

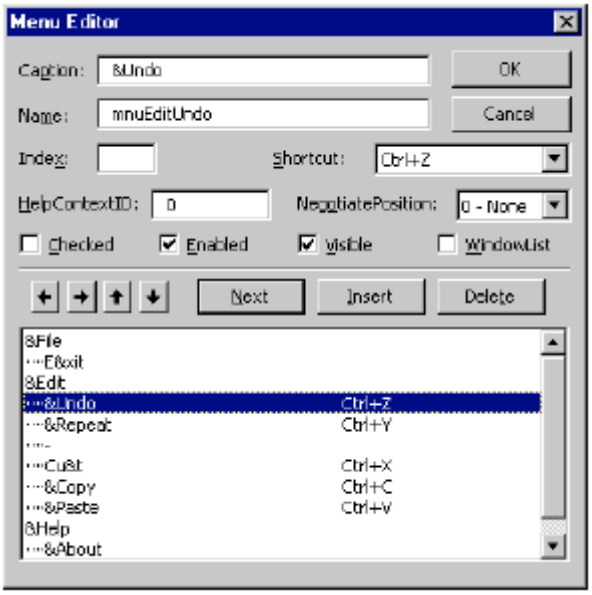


Figura 5.1. Editor de menús de *Visual Basic*.

Entre las capacidades de ***Visual Basic 6.0*** no podía faltar la de construir menús con gran facilidad. Sin embargo, hay algunas diferencias respecto al modo el que se construyen los controles. Para crear menús ***Visual Basic*** dispone de una herramienta especial que se activa mediante el comando ***Menu Editor*** del menú ***Tools***. El cuadro de diálogo que se abre se nuestra en la **Figura 5.1.** Más adelante se verá cómo se utiliza esta herramienta; antes, conviene recordar brevemente las características más importantes de los menús de ***Windows.***

Los menús presentan sobre los demás controles la ventaja de que ocupan menos espacio en pantalla, pero tienen el inconveniente de que sus posibilidades no están a la vista más que cuando se despliegan.

5.1 INTRODUCCIÓN A LAS POSIBILIDADES DE LOS MENÚS

La mayor parte de las aplicaciones de ***Windows*** utilizan menús. Aunque todo el mundo está familiarizado con sus funciones más básicas, conviene ver algunas posibilidades menos usuales. Se utilizarán para ello unas aplicaciones tan conocidas como ***Word*** y ***Excel***.

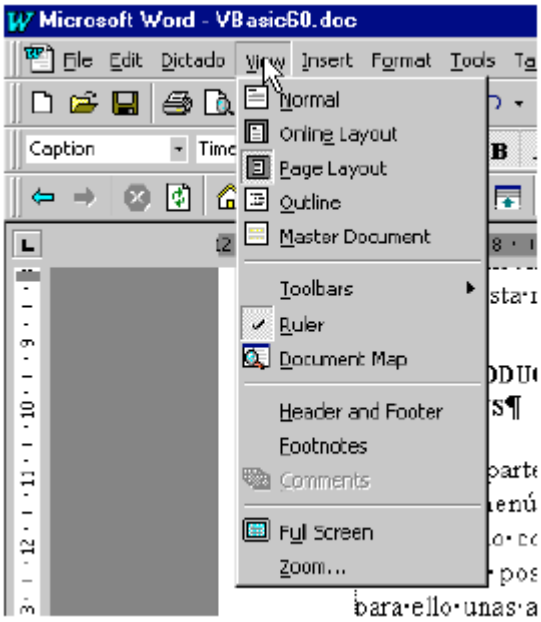


Figura 5.2. El menú *View* de *Word 97*.

La **Figura 5.2** recoge el aspecto del menú ***View*** de ***Word 97***, en el que conviene destacar las siguientes características:

1. Lo primero que llama la atención es que los menús aparecen divididos en grupos de Opciones separados por líneas horizontales.

2. Algunos items como ***Page Layout*** tienen un icono resaltado a su izquierda. Esto quiere decir que ese ítem es la opción elegida entre los cuatro items de su grupo. En este sentido los menús se parecen a los controles ***OptionButton***. ***Visual Basic 6.0*** no permite hacer esto directamente, pero lo puede simular.

3. Otros items como ***Ruler*** tienen una marca de selección a su izquierda. En este caso el Menú realiza la función de las cajas de selección (***CheckBox***).

.4. Todas las opciones del menú tienen una letra subrayada. La finalidad es poder desplegar y activar los menús desde teclado, sin ayuda del ratón (con ***Alt*** y la letra subrayada).

5 También se observa que el ítem ***Comments*** aparece en gris claro. Esto quiere decir que en estemomento no está activo y por tanto no es seleccionable.

6. Otros items como ***Toolbars*** están seguidos por un pequeño triángulo. Eso quiere decir que existe un menú secundario con más opciones. Otros items como ***Zoom*** aparecen seguidos por puntos suspensivos (...). Este es un convenio utilizado para indicar que eligiendo esa opción seabrirá un cuadro de diálogo en el que habrá que tomar otras decisiones.

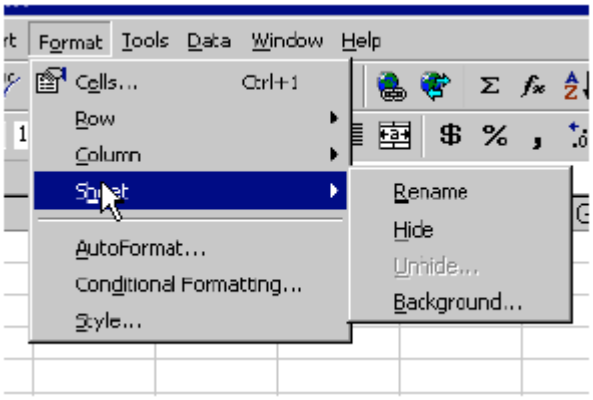


Figura 5.3. El menú *Format/Sheet* de *Excel 97*.

Por lo que respecta al menú de ***Excel 97*** que aparece en la Figura 5.3 la característica más importante es que tiene ***sub-menús*** (señalados mediante un pequeño triángulo a su derecha), que se abren al colocar el cursor sobre el ítem correspondiente. Estos menús se suelen llamar ***menús en cascada***, y son muy frecuentes en ***Windows.***.

Otra característica de los menús, que no aparece en las **Figura 5.2** y **Figura 5.3**, es la posibilidad de definir combinaciones de teclas que realizan la misma función que una opción del menú. Por ejemplo, en muchas aplicaciones ***Ctrl+C*** equivale a ***Edit/Copy*** y ***Ctrl+V*** a ***Edit/Paste***.
Estas combinaciones de teclas se llaman *accesos rápidos* (***shortcut***) y hay que distinguirlas de acceder a los menús mediante la tecla ***Alt*** y las letras subrayadas de los nombres.

Continuará.....

Nota de Radacción: El lector puede descargar este capítulo y capítulos anteriores del curso desde la sección ***"Artículos Técnicos"*** en el sitio web de **EduDevices (www.edudevices.com.ar)**