

# CUADERNO DE TRABAJO VISUAL BASIC



Cuaderno de Trabajo Visual Basic se encuentra bajo una Licencia [Creative Commons Atribución-No Comercial-Licenciamiento Recíproco 3.0 Unported](https://creativecommons.org/licenses/by-nc-sa/3.0/). Septiembre 2011 – IDSystems

# Contenido

INTRODUCCION .....	5
Actividad de Aprendizaje 1. ....	6
Actividad de Aprendizaje 2. ....	7
Actividad de Aprendizaje 3 .....	8
Actividad de Aprendizaje 4 .....	8
Ejercicio 1 – Uso de aplicaciones visuales.....	9
Ejercicio 2 – Iniciando con Visual Basic.....	10
Ejercicio 3 – Identificación de elementos del Entorno IDE.....	12
Ejercicio 4 – Primera aplicación .....	16
Ejercicio 5 – Introduccion al codigo .....	21
Ejercicio 6 – Guardar el formulario y el proyecto .....	26
Ejercicio 7 – Unidades de Temperatura.....	27
Ejemplo 8 – Colores y posiciones.....	29
Ejercicio 9 – MiniCalculadora.....	32
PRACTICA 1- Preguntando tu nombre .....	34
PRACTICA 2 – Preguntando tu nombre (Print) .....	34
Ejercicio 10 – Calculadora sencilla .....	35
<b>Propiedades</b> del formulario .....	35
<b>Añadir</b> objetos al formulario.....	39
<b>Modificar</b> propiedades de varios objetos simultáneamente .....	41
<b>Fuentes</b> de letra en modo edición. ....	42
<b>Fuentes</b> de letra en modo ejecución .....	44
<b>Tamaño</b> automático.....	45
<b>Alineación</b> del texto .....	45
<b>Delimitación</b> de tamaño .....	47
<b>Texto</b> de ayuda .....	47
<b>OptionButton</b> en modo gráfico.....	48
Ejercicio 11 – Declaracion de variables.....	49
Ejercicio 12 – Declaracion de variables explicitas.....	50
Ejercicio 13 – Variables .....	51
Ejercicio 14 – Declarando Constantes .....	53
PRACTICA 3 – Convertir Horas .....	55

PRACTICA 4 – Calculo de Areas .....	56
PRACTICA 5 – Botones de opcion y colores 2 .....	57
Actividad de Aprendizaje 5 – Funciones Matematicas (L) .....	58
Ejercicio 15 – Calculo de Salario Neto (Formulas) .....	59
Ejercicio 16 – Restaurante (Formulas) .....	65
Ejercicio 17 – Calculo de Edad .....	73
PRACTICA 6 – Cajero Automatico – Desglose de billetes .....	75
Ejercicio 18 – If...Then..Else .....	76
Ejercicio 19 – If..Then.....	77
Ejercicio 20 – If... Anidados .....	78
PRACTICA 7 – Configura computadora de compra.....	79
Ejercicio 21 – Boleta de pago (Condiciones If).....	79
Ejercicio 22 – Puntuacion de un estudiante (Condiciones If) .....	86
Ejercicio 23 – Fechas en modo completo .....	91
PRACTICA 8 – Descuento en Restaurante.....	95
Ejercicio 24 – Select Case.....	95
Ejercicio 25 – Select Case con operadores de comparacion .....	96
Ejercicio 26 – If..Then y Select Case.....	96
Ejercicio 27 – For...Next .....	100
Ejercicio 28 – For...Next con validacion numerica .....	103
Ejercicio 29 – Do...Loop.....	104
Ejercicio 30 – Tablas de multiplicar .....	110
Ejercicio 31 – Numeros primos .....	112
Ejercicio 32 – Cifras y divisores de un numero .....	114
PRACTICA 9 – Centro numerico en una lista.....	121
Ejercicio 33 – Funciones o Procedimientos .....	121
Ejercicio 34 – Matrices de Controles .....	124
Ejercicio 35 – Ordenacion por burbuja .....	125
Ejercicio 36 – Pedido de cotizaciones .....	130
Ejercicio 37 – Ficha de matricula .....	135
PRACTICA 10 – Consulta de cursos .....	143
Ejercicio 38 – Concatenacion. Funciones de cadenas.....	143
Ejercicio 39 – Eventos .....	145

Ejercicio 40 – Menu.....	147
<b>Editor</b> de menús .....	147
<b>Título</b> de menú.....	148
<b>Interior</b> de un menú.....	150
<b>Introducir código</b> en los menús.....	152
<b>Activar y desactivar</b> menús.....	153
<b>Líneas de separación</b> .....	155
<b>Creación</b> de submenús .....	156
<b>Marca de verificación</b> .....	158
<b>Activar y desactivar</b> Submenús.....	160
<b>Borrar lista</b> .....	161
<b>Borrar elemento</b> .....	162
<b>Teclas de método abreviado</b> .....	163
<b>Menú contextual</b> .....	164
Ejercicio 41 – Cadena invertida.....	166
Ejercicio 42 – Ficha de matricula (con varios forms) .....	168
Ejercicio 43 – Tipos de Formularios (MDI) .....	174
PRACTICA 11 – Reserva de agua .....	178

# INTRODUCCION

Este Cuaderno de Trabajo esta basado en varios cursos presenciales, online, de centros de capacitacion y universidades encontrados en la red, y durante el ejercicio de la profesion como docente. Hay muchisimos ejemplos y practicas en Internet actualmente sobre esta materia de Visual Basic, sin embargo, se han seleccionado solamente aquellos que son mas didacticos y estan enfocados en los temas de aprendizaje de Visual Basic como curso o materia.

Algunas de las practicas son en parte ejercicios porque contienen algo de codigo para facilitar la elaboracion de dichos programas y el aprendizaje.

Recordemos que este es solo un cuaderno de trabajo que no sustituye a su Guia de Estudio (teoria) que lleve durante el curso, sino que lo complementa para que al realizar dichos programas su aprendizaje sea mas completo.

Encontrara:

Ejercicios. Son ejemplos paso a paso, explicados totalmente, con las características y propiedades de cada control, como hacerlo y el código a teclear. Generalmente los ejercicios traen anexo un cuestionario de preguntas para reforzar el conocimiento adquirido.

Practicas. Son ejemplos de problemas que es imperativo que el alumno desarrolle por si mismo. No se proveen soluciones.

Actividades de Aprendizaje. Son ejercicios o practicas paso a paso con mediana solucion, dejando alguna parte para desarrollar por parte del alumno.

Proyectos. Son ejemplos completos que involucran muchos temas vistos a lo largo del curso y estan como muestra de lo que es capaz de hacer Visual Basic

Agradecimientos:

AreaInteractiva (desaparecida)

Carlos Castillo

Mirta Echeverria

Manuel Perez Valdes

Y otros

## Actividad de Aprendizaje 1.

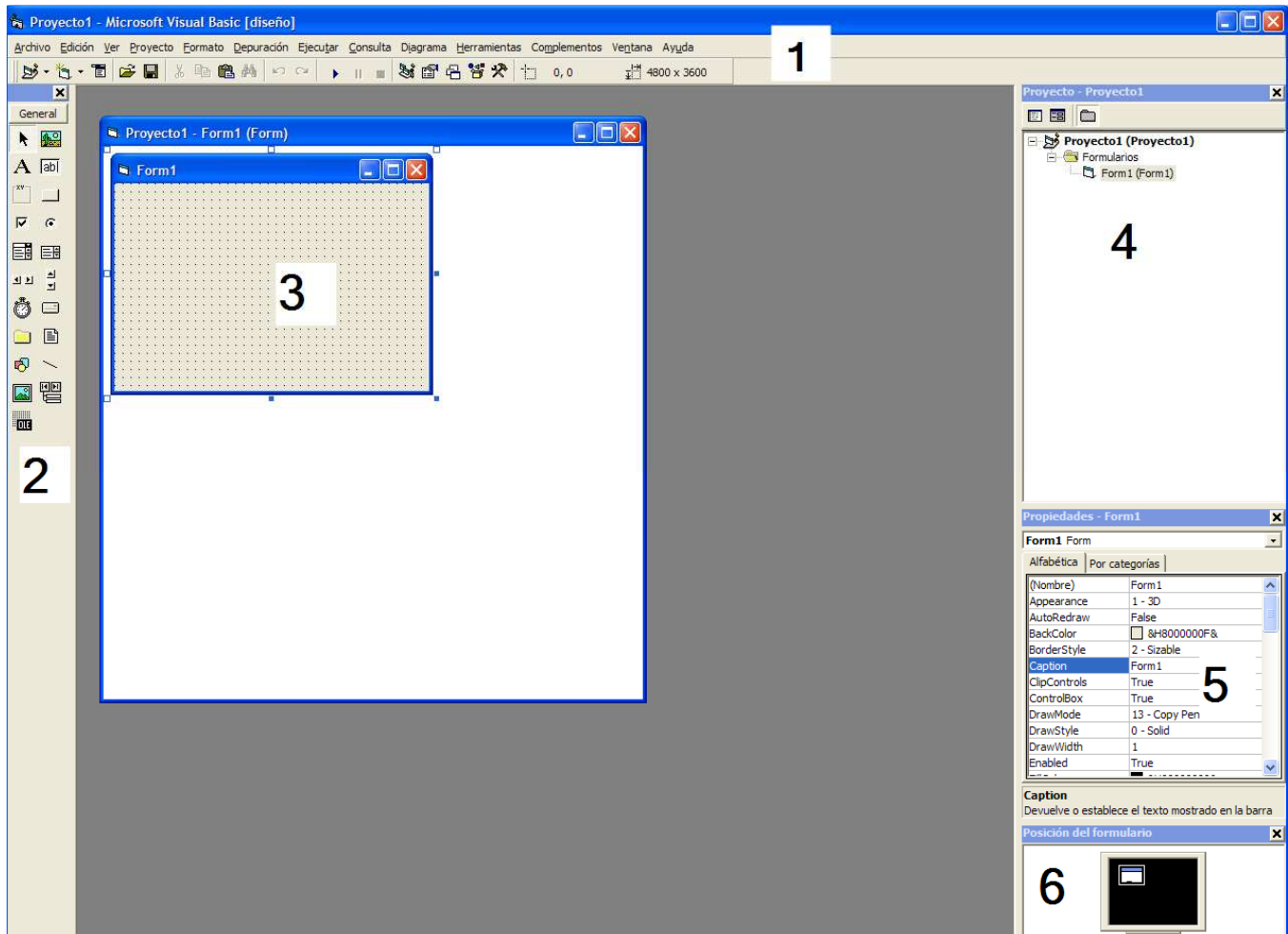
Instrucciones: Realizaras un glosario simplificado de terminos basicos con el que trabajarás en Visual Basic, basandote en lo abordado durante tus primeras clases.

Los conceptos a definir son:

- Programa secuencial
- Programa ordinario
- Modo de diseño
- Modo de ejecución
- Formulario
- Objetos
- Propiedades
- Nombre de objetos
- Eventos
- Metodos
- Proyecto
- Fichero o Archivo.

## Actividad de Aprendizaje 2.

De acuerdo a lo aprendido hasta el momento, revisando la lectura de tu guía y explicación del profesor, identifica los elementos que comprende el entorno de Visual Basic.



	ELEMENTO	DESCRIPCION
1		
2		
3		

4		
5		
6		

### Actividad de Aprendizaje 3

Realizaras una investigacion sobre los lenguajes de programacion mas utilizados en la actualidad. De cada uno de ellos deberas reportar:

1. Nombre
2. Tiempo de operación
3. Caracteristicas principales
4. Usos frecuentes

### Actividad de Aprendizaje 4

Agrega a tu glosario los siguientes terminos y su descripcion.

1. Comentarios
2. Sentencias
3. Variables y sus tipos
4. Procedimientos



## Ejercicio 1 – Uso de aplicaciones visuales

1. Abre la calculadora
2. Pulsa sobre el boton 6

De esta forma podemos ver que el contenido del boton (el numero 6) a pasado al cuadro de texto donde iran apareciendo las cantidades y resultados de nuestras operaciones, pero antes de esto se ha borrado el 0 que estaba en este recuadro de texto.

3. Vuelve a pulsar el boton 6

Fijate en lo que ha pasado ahora. El nuevo 6 no ha sustituido (borrado) lo que habia en el cuadro de texto, sino lo que ha hecho la calculadora es poner el segundo 6 seguido del primero con lo que tenemos el numero 66.

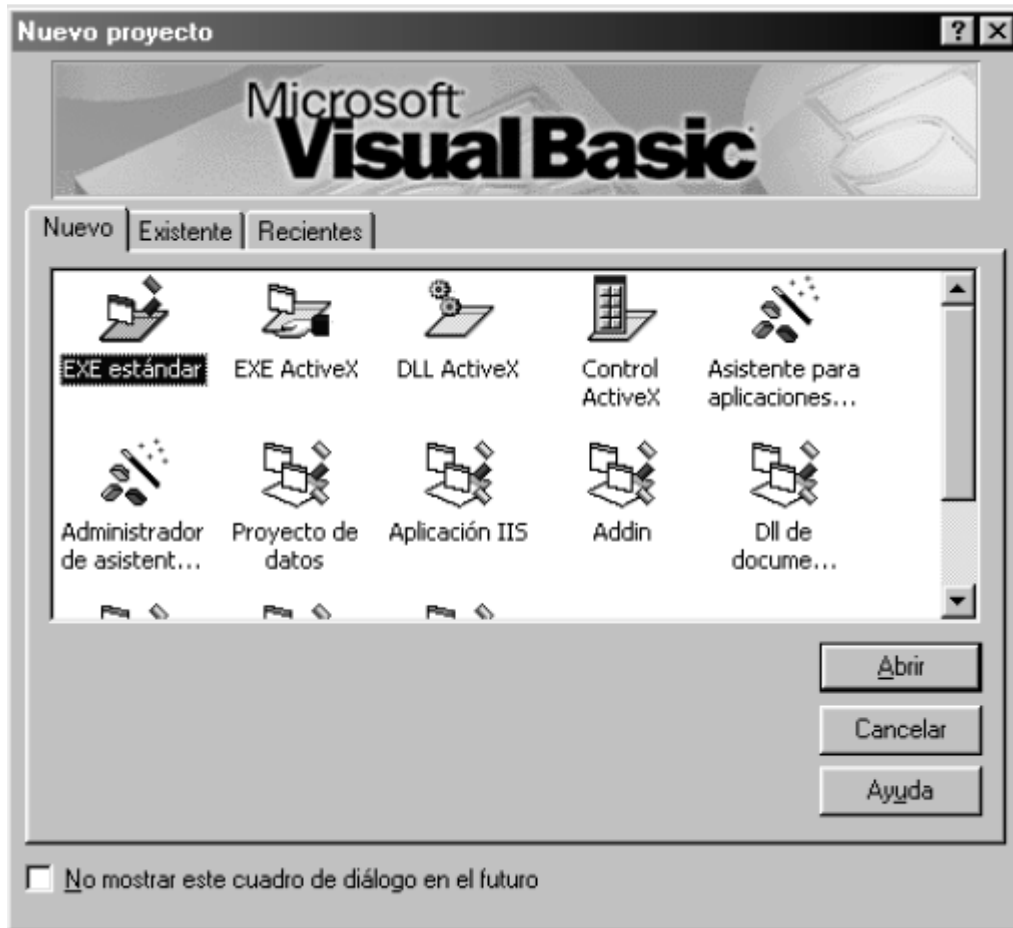
Con esto podemos ver que el boton 6 ha actuado de dos formas diferentes, aunque nosotros lo hallamos activado igual. ¿Por qué el boton 6 ha actuado asi? Pues por la simple razon que el boton antes de actuar ha mirado a su alrededor y según lo que ha visto ha reaccionado de una forma u otra. Al decir que mira a su alrededor queremos decir que mira que propiedades y características cumplen los otros elementos que forman parte de la aplicación.

Pues bien, nosotros como buenos programadores deberemos tener en cuenta que es lo que nos interesa que realice un objeto en cada momento determinado y como queremos que lo realice. Para que esto sea asi nos debemos plantear cuando, como y porque el usuario realizara un evento y como debe actuar este.

Debemos pensar que este punto, junto con la comunicación con el usuario (ya hablaremos mas adelante), son dos de los puntos mas importantes dentro de la programacion al estilo de Visual Basic.

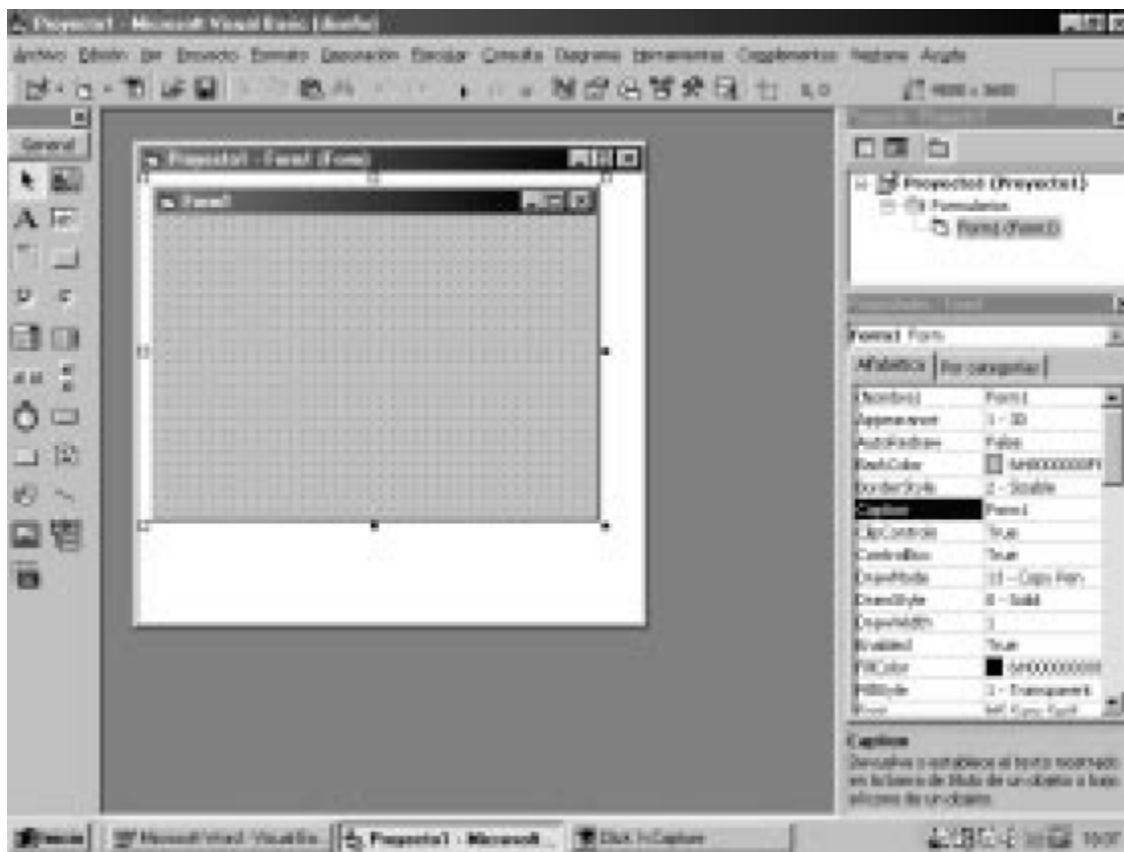
## Ejercicio 2 – Iniciando con Visual Basic

1. Inicia Visual Basic: Inicio – Programas – Microsoft Visual Studio 6.0 – Microsoft Visual Basic 6



Al Iniciar Visual Basic te aparecera en primer termino una pantalla como esta:

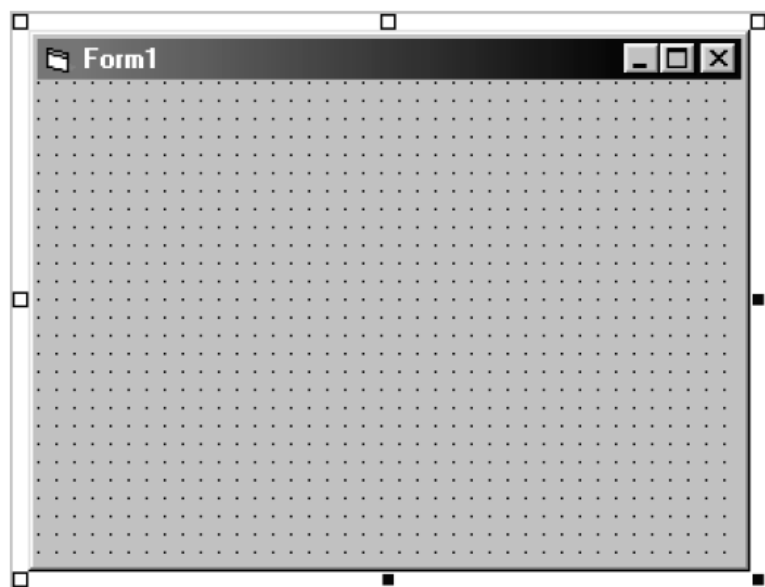
2. Haz un clic en Aceptar para iniciar un nuevo proyecto.  
Observa la siguiente pantalla e identifica las partes que iremos nombrando a continuacion.



## Barras de menus

En las barras de menus tenemos todas las opciones, utilidades y comandos de los que podemos disponer en Visual Basic (Archivo, Edicion, Ver, etc.)

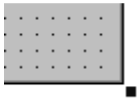
## Formulario



Esta es una de las partes mas importantes, ya que aquí es donde diseñaremos la pantalla o pantallas que formaran parte de nuestro programa. A estas pantallas le llamaremos formularios. Aquí iremos “pegando” y modificando los diferentes elementos de nuestra aplicación, como puedan ser botones, cuadros de texto, etc. Si no vieramos la pantalla del formulario podriamos activarla desde **Ver – Objeto** o pulsar **Mayusculas + F7**.

El diseño de una pantalla es tan simple como arrastrar los objetos que deseamos, desde el **cuadro de herramientas** hasta el **formulario**. Para modificar el tamaño de cualquier objeto, incluso del **formulario** solo es necesario situarse en cualquiera de las esquinas del objeto o en el centro de uno de sus lados marcados con un cuadrado, esperar que el raton se convierta en una flecha de desplazamiento, pulsar el boton izquierdo del raton y mientras se mantiene pulsado movernos hasta que el objeto tome un nuevo tamaño. Si cambiamos el tamaño desde uno de los vertices podremos modificar tanto el alto como el ancho, mientras que si arrastramos desde uno de los lados solo podremos modificar el alto o el ancho dependiendo del lado en el que nos encontremos.

### Ejercicio 3 – Identificacion de elementos del Entorno IDE



- 1.- Sitúate sobre la esquina inferior derecha del **formulario**, sobre el cuadrado pequeño inferior.
- 2.- Espera hasta que el ratón se convierta en una doble flecha, pulsa y arrastra hasta que veas como el **formulario** cambia de tamaño.

Así de fácil.

#### **Cuadro de herramientas**

En este cuadro encontramos las herramientas que podemos utilizar para diseñar nuestro proyecto. El **cuadro de herramientas** que presentamos a continuación es el estándar, el cual contiene los elementos básicos. Más adelante veremos como podemos agregar elementos a este **cuadro de herramientas**.

A continuación vamos a nombrar las herramientas básicas, para así poder empezar a crear una pequeña aplicación. En futuras lecciones iremos explicando el resto de herramientas.



**Puntero.** Utilizaremos este control para poder mover, cambiar el tamaño o seleccionar los diferentes elementos que insertemos en el formulario.



**Label.** Utilizaremos este control para escribir etiquetas donde aparecerá texto que el usuario no podrá cambiar.



**TextBox.** Son cuadros de texto que el usuario podrá cambiar.



**CommandButton.** Utilizaremos este control para crear botones sobre los cuales podrá actuar el usuario.



**CheckBox.** Casilla que el usuario podrá utilizar para marcar dos posibles opciones. Verdadero o falso, sí o no, activado, desactivado... El usuario podrá marcar la cantidad de casillas de verificación que desee dentro de una aplicación.



**OptionButton.** Muy parecida al control anterior, pero el usuario solo podrá marcar una de las opciones. Si tenemos dos controles de este tipo, en el momento de seleccionar uno automáticamente se quitará la selección el otro.



Para visualizar el **cuadro** de herramientas podremos ir a la opción **Cuadro de herramientas** dentro de la opción **Ver** o hacer un clic en este botón: en la barra de herramientas (definida a continuación).

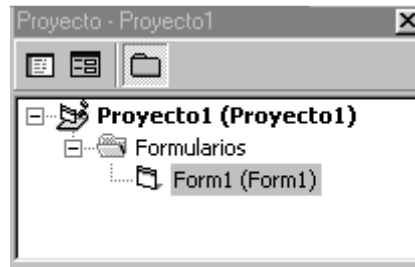
### Barra de herramientas

Desde las **barras** de herramientas podemos acceder a todas aquellas instrucciones o comandos que son usados cuando estamos editando y programando nuestra aplicación (**Grabar, abrir, ejecutar, mostrar** diferentes elementos de **Visual Basic, etc.**). Al iniciar **Visual Basic** aparece una **barra** de herramientas estándar. Nosotros podemos ocultar o mostrar otras barras de herramientas, las cuales ya veremos.



Para visualizar la **Barra** de herramientas **estándar** debemos ir a la opción **Barra de herramientas dentro** de la opción **Ver**. Allí podremos encontrar diferentes **Barras** de herramientas para que se active una de ellas solo deberás hacer un clic sobre el nombre deseado. En este caso haríamos un clic sobre **Estándar**.

## Explorador de proyectos



Desde el **explorador** de proyectos podemos ver todas “las pantallas”, formularios, que componen nuestra aplicación.



Para poder visualizar el **explorador** de proyectos deberás ir a **Ver – Explorador** de proyectos, pulsar la combinación de teclas **Ctrl + R** o pulsar sobre este botón: en la barra de herramientas.

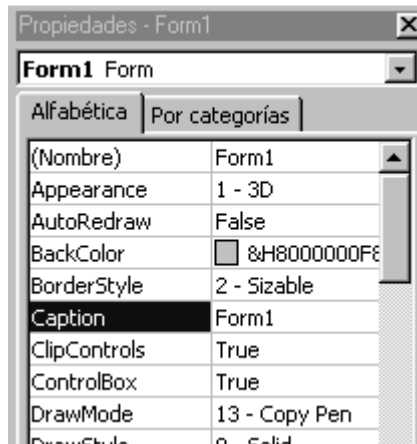
### Ventana de propiedades

En esta pantalla vemos las **propiedades** de los objetos que tenemos seleccionados. (Las **propiedades** las veremos con más detenimiento en futuras lecciones). Las **propiedades** son las características que puede tener cada uno de los elementos como puede ser su tamaño, su posición, su contenido, su color, su forma, su tipo de letra, etc. Todas estas propiedades se pueden cambiar cuando nos encontramos en forma **diseño**, creando el programa, o en forma **ejecución**, cuando estamos ejecutando la aplicación.

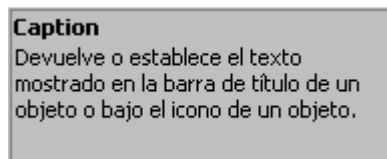
Para cambiar una propiedad de un objeto cuando estamos en modo diseño, solo tenemos que seleccionar el **objeto** ir a la **ventana** de propiedades y cambiar la **propiedad** que nos interese. Más adelante realizaremos unas cuantas prácticas donde veremos como hacerlo.



Si no nos aparece la **ventana** de propiedades podemos pulsar **F4**, o ir a la opción de la barra de menús **Ver – Ventana** propiedades o como última opción utilizar el botón de la **barra** de herramientas:



Observa que en la parte inferior de la **ventana** de propiedades aparece un pequeño cuadrado en el que tienes una pequeña ayuda sobre la propiedad seleccionada.



Los demás elementos que aparecen en tu pantalla los iremos comentado en siguientes lecciones.

## Ejercicio 4 – Primera aplicación

Vamos a realizar una pequeña aplicación donde podremos empezar a utilizar todo lo que hemos visto hasta el momento. Si alguna de las cosas que explicamos no te queda del todo clara, no te preocupes, ya lo irás entendiendo a medida que avances en el curso. Lo importante de esta práctica es crear una primera aplicación donde veas el funcionamiento de diferentes objetos y las propiedades de estos. Así que sin más demora, adelante y sin miedo.

1. Inicia **Visual Basic 6.0**.
2. De la pantalla **Nuevo proyecto** escoge la opción **EXE estándar** y pulsa **Aceptar**.

Después de unos segundos tendrás en pantalla un nuevo **formulario**, donde crearemos nuestra primera aplicación.

### Tamaño del formulario

3. Pulsa un clic sobre el **formulario**, observa como en el cuadro de las propiedades aparece el nombre del **formulario**, que por defecto es **Form1**.
4. Busca la propiedad **Height** (Las propiedades están ordenadas alfabéticamente).
5. Haz doble clic sobre esta propiedad y escribe **3100**. Pulsa **Intro**.

Observa como el ancho de nuestro formulario ha cambiado.

6. Busca la propiedad **Width**.
7. Haz doble clic sobre dicha propiedad y escribe **4300**.

Ahora podemos observar como la altura de nuestro formulario ha cambiado. Vamos ha empezar a colocar los elementos necesarios para que funcione nuestra aplicación. De tal forma que queden como en la siguiente imagen. (Sigue los pasos que te indicamos, no te avances)



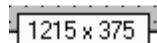


8. Colócate con el puntero del ratón en el **cuadro** de herramientas sobre del control **CommandButton**.

9. Pulsa un doble clic sobre este control, verás como ha aparecido un **botón** en el centro de nuestro **formulario**.

**Cambio del tamaño del botón**

10. Sitúate sobre la esquina inferior derecha de dicho elemento.



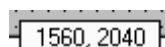
11. Mantén pulsado el ratón hasta que aparezca el siguiente recuadro:

(Puede ser que los valores de tu recuadro no sean iguales que los que aparecen en esta imagen). Este recuadro nos informa del ancho y alto del objeto.

12. Muévete, arrastrando hasta que dentro del recuadro aparezca **1215 x 375**. Cuando consigas estos valores suelta el botón del ratón.

Fíjate en las siguientes propiedades y sus valores dentro del cuadro de propiedades **Height = 375** y **Width = 1215**.

**Cambio de posición de un objeto.**



13. Haz un clic sobre el botón que acabamos de crear. Después de unos instantes te aparecerá un recuadro con dos números separados con una coma.

Este recuadro nos indica la posición que se encuentra el elemento con respecto a la esquina **izquierda superior** de nuestro **formulario**.

14. Mantén pulsado el botón del ratón y muévete hasta la posición **1560**, 2040 aproximadamente.

Ahora observa los valores de las propiedades **Top = 2040** y **Left = 1560**.

Es importante que recuerdes para que se utilizan las propiedades: **Height**, **Width** y **Top**, **Left**.

#### Cambio del nombre del botón

La propiedad (**Nombre**), nos servirá para referirnos a este objeto en el momento que estemos programando.

15. Selecciona el **botón** haciendo un clic sobre él. Pulsa **F4**. Este punto es solo necesario en caso de no tener el botón seleccionado.

16. Haz un doble clic en la propiedad (**Nombre**), (está situada en la primera posición).

17. Escribe **Copiar**. Pulsa **Intro**.

A partir de este momento siempre que queramos hacer referencia al botón de nuestro formulario utilizaremos el nombre **Copiar**.

#### Cambio del texto del botón.

Ahora, para que el usuario de nuestra aplicación tenga un poco de idea que hace este botón vamos a cambiar su texto.

18. Vuelve a pulsar **F4**.

19. Haz un doble clic sobre **Caption** y escribe **C&opia**

El signo **&** delante de la **o** nos marcará la combinación de teclas que podremos utilizar para que se active nuestro botón. En este caso sería **Alt+o**. Observa como dentro del botón aparece escrito **Copia**.

Vamos a colocar los demás elementos que forman parte de nuestra aplicación.

#### Creación de un TextBox

20. Pulsa doble clic sobre el **TextBox**.

21. Colócalo utilizando el método que quieras dentro del **formulario** en la posición **240**, 240 con un tamaño de **1455 x 285**.

22. Cambia la propiedad (**Nombre**) por **Texto**.

23. Sitúate sobre la propiedad **Text** y borra el contenido.

De esta forma haremos que cuando iniciemos el programa no aparezca ningún texto en el interior de este objeto.

#### Creación de un Label

24. Coloca un **Label** en la posición 2280, 240 con un tamaño de 1575 x 255.

25. Cambia su nombre por **Etiqueta**.

26. Sitúate sobre la propiedad **Caption** y borra el contenido.

De esta manera haremos que cuando ejecutemos la aplicación no exista ningún texto dentro de este objeto.

Fíjate que para cambiar el contenido del objeto **TextBox** utilizamos la propiedad **Text**, mientras que en el objeto **Label** utilizamos **Caption**.



27. Sitúate sobre la propiedad **BorderStyle** del **Label**. Abre la lista desplegable de la misma propiedad y escoge la opción **1-Fixed Single**.

Con esta opción lo que conseguimos es que el **Label** tenga un borde, con el que podemos ver el límite de este control.

#### Creación de CheckBox

Vamos a colocar dos **CheckBox**, con los que controlaremos si queremos el texto en **Negrita**, **Cursiva** o las dos cosas. Recuerda que los controles **CheckBox** pueden estar los dos activados, uno solo, o los dos desactivados.

28. Pulsa doble clic sobre el **CheckBox** del **Cuadro** de herramientas.

29. Sitúalo en la posición 600, 840

30. Coloca otro **CheckBox** en la posición 600, 1200

31. Cambia el nombre del primero por: **Negrita** y al segundo **Cursiva**.

32. Cambia el **Caption** del primero de ellos por **Negrita** y el segundo por **Cursiva**. Observa cual será en cada caso la tecla que activará este objeto.

Fíjate en la imagen del principio de la práctica para ver como han de quedar los controles.

### Creación de **OptionButton**

Ahora colocaremos dos **OptionButton**, con estos nuevos controles podremos controlar si lo que queremos es que aparezca todo el texto en **Mayúsculas** o en **minúsculas**. Utilizamos este tipo de control ya que solo podemos hacer que el texto aparezca todo en mayúsculas o todo en minúsculas.

33. Pulsa doble clic sobre el **OptionButton** del **Cuadro** de herramientas.

34. Sitúa el primer **OptionButton** en la posición: 2280, 840 y el segundo en la posición: 2280, 1200

35. Cambia el **nombre** de los dos controles por **Mayusculas**, el primero y **Minusculas**, el segundo.

Observa que en el nombre no hemos puesto acentos. Podríamos ponerlos pero hay que pensar que muchos lenguajes de programación no los aceptan.

36. Cambia el **Caption** de ambos por Mayúsculas y Minúsculas.

Fíjate en la imagen del principio de la práctica para ver como han de quedar los controles.

El tamaño de estos controles no lo controlamos ya que los bordes de estos elementos no se ven en el modo de ejecución.

### Cambio del título e icono del formulario.

37. Selecciona el formulario.

38. Accede a la propiedad **Caption** y escribe: **Primer programa**.

Veras que mientras lo escribes aparece en el título del formulario.



39. Ahora accede a la propiedad **Icono** y pulsa en este botón

Te aparecerá una ventana típica de **Windows** para búsqueda de archivos.

40. Accede al directorio donde tienes instalado **Visual Basic**. Selecciona el archivo **Trffc14.ico** que se encuentra dentro del siguiente directorio **Graphics\Icons\Traffic**

Acto seguido aparecerá un icono en el formulario.

Perfecto, ya tenemos colocados todos los elementos que forman parte de nuestra primera aplicación. Ahora solo nos queda completar el código con el cual la aplicación realizará su cometido.

## Ejercicio 5 – Introduccion al codigo

¿Dónde colocaremos el código de nuestra aplicación? En esta aplicación es muy fácil saber, ya que tenemos que colocar el código allí donde al realizar un evento se produzca una “reacción”. Bien, en nuestro caso queremos que se realice cuando pulsemos el botón Copiar.

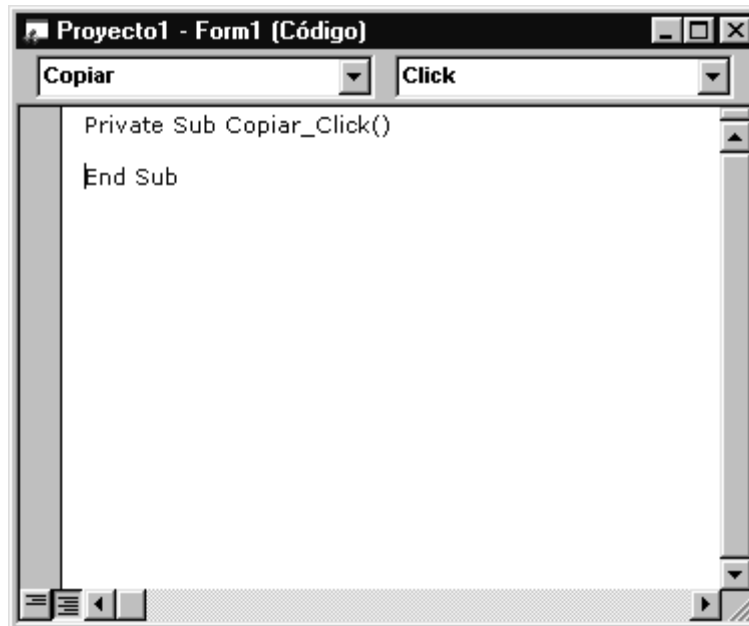
Tenemos que pensar que cada evento podrá tener una serie de instrucciones que se ejecutarán cuando éste se produzca. A este grupo de instrucciones dentro de un evento le llamaremos **procedimiento** de evento. Cada **procedimiento** de evento se distingue de otro porque aparece el nombre del control (Nombre), más un carácter **\_** y el nombre del evento. Por ejemplo **Boton\_Click**, indica que el procedimiento se ejecutará cuando se hace un clic sobre el botón llamado **Boton**.

Nosotros desde el interior de un **procedimiento** podemos cambiar la propiedad de cualquier elemento que exista en nuestro formulario. Esto lo haremos indicando el **nombre** del objeto al que queremos cambiar la propiedad seguido de un punto (.) y el **nombre** de la propiedad a cambiar. Por ejemplo **Etiqueta.Caption** = “Cambio de texto”, con esto cambiaríamos el **Caption** de un **Label** llamado **Etiqueta** haciendo que aparezca “Cambio de texto”. En lecciones posteriores veremos con mucho más detenimiento las instrucciones y comandos de **Visual Basic**.

En nuestro ejemplo queremos que al pulsar el botón **Copiar** el ordenador copie en el **Label** el texto que hay en el **TextBox** con los formatos que indique los demás elementos: **Mayúsculas o minúsculas, Negrita, Cursiva**.

1. Pulsa doble clic sobre el botón **Copiar**.

Acto seguido aparecerá una ventana como esta:



En esta ventana será donde nosotros introduciremos el código que queremos que realice nuestro procedimiento.

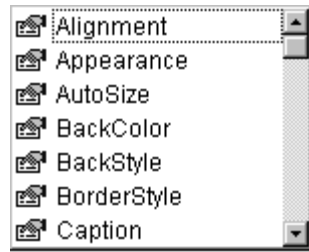
El código deberá estar entre las dos líneas que aparecen ya escritas, ya que estas nos indican el principio y el final de dicho procedimiento de evento.

La primera línea nos indica que estamos programando dentro del evento Click (hacer un clic con el ratón) dentro del objeto Copiar. Y la segunda línea nos indica el final de dicho procedimiento de evento.

Antes de empezar a copiar el código que irá en este botón explicaremos una “herramienta” que forma parte de Visual Basic que nos facilita un poco el trabajo y nos ayuda a la hora de escribir el código.

Vamos a introducir una primera línea de código poco a poco para ver que es lo que ocurre.

2. Escribe lo siguiente: **Mayusculas.**



Observa como acto seguido de poner un punto te aparece una especie de menú contextual parecido a este:

En este menú contextual han aparecido todas las propiedades del objeto **Mayusculas**.

3. Escribe **v**.

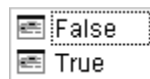
Observa como la lista ha saltado hasta encontrar la primera palabra que empezaba con **V**.

4. Pulsa la tecla **Tab**.

Observa como automáticamente ha aparecido escrito en pantalla **Value**.

5. Escribe **=**

Acto seguido aparece otro menú contextual con solo dos opciones:



6. Escribe **T** (es igual en minúsculas que en mayúsculas).

7. Pulsa **Intro** ya que hemos llegado al final de la línea.

Observa como **Visual Basic** coloca los espacios en los lugares correspondientes. Si **Visual Basic** hubiera encontrado algún error de escritura nos lo hubiera hecho saber con un mensaje de error y poniendo la línea en color rojo.

Cuando empieces a escribir el código podrás ver que según que tipo de instrucción introduzcas **Visual Basic** te ofrecerá otra especie de menú contextual con la estructura de esta instrucción. Este será el caso, por ejemplo, de la instrucción **UCase** que escribirás en las siguientes líneas de código.

8. Borra la línea de código que has escrito.

No borres las dos líneas de las que hemos estado hablando en el principio de este capítulo.

9. Copia el siguiente código, entre las líneas que te hemos indicado al principio de este capítulo:

```
Etiqueta.Caption = Texto.Text
If Negrita.Value = 1 Then
    Etiqueta.Font.Bold = True
Else
    Etiqueta.Font.Bold = False
End If
If Cursiva.Value = 1 Then
    Etiqueta.Font.Italic = True
Else
    Etiqueta.Font.Italic = False
End If
If Mayusculas.Value = True Then
    Etiqueta.Caption = UCase(Etiqueta.Caption)
Else
    Etiqueta.Caption = LCase(Etiqueta.Caption)
End If
```

Vamos a comentar un poco las líneas que hemos utilizado en nuestra aplicación:

**Etiqueta.Caption = Texto.Text** ‘Copiamos el contenido de la casilla de texto **Texto.Text** a la etiqueta **Etiqueta.Caption**

**If Negrita.Value = 1 Then** ‘Con la instrucción **If** hacemos una pregunta que el ordenador nos contestará con **Verdadero** o **Falso**. (Esta estructura la veremos con mucho más detenimiento en lecciones futuras pero ahora adelantamos la estructura para que sea más fácil el entendimiento del código).

```
If [Pregunta] Then
    [Instrucciones cuando la Pregunta es verdadera]
    ...
Else
    [Instrucciones cuando la Pregunta es falsa]
    ...
End If
```

En nuestro caso preguntamos si el **CheckBox** llamado **Negrita** está activado. Esto lo hacemos con la propiedad **Value** que solo puede tener dos valores **1** = activado o **0** = desactivado.



En el caso que la casilla **Negrita** esté activada (**Value = 1**), primera parte de la instrucción **If**, entonces el contenido de la **Etiqueta** se pondrá en **Negrita** poniendo la propiedad **Etiqueta.Font.Bold** a **Verdadero (True)** de la siguiente manera: **Etiqueta.Font.Bold = True**

En caso que la casilla **Negrita** no esté activada (**Value = 0**), segunda parte de la instrucción **If**, entonces el contenido de la **Etiqueta** no aparecerá en **negrita**, poniendo la siguiente instrucción **Etiqueta.Font.Bold = False**

En el siguiente **If** lo que hacemos es mirar si el **CheckBox** llamado **Cursiva** está activado. Si está activado pondremos la propiedad **Etiqueta.Font.Italic** a **verdadero (True)**, mientras que si no está activado, **Else**, pondremos en valor a **falso (False)**.

**If** **Mayusculas.Value = True** Then ‘Con este otro **If** lo que hacemos es mirar si el **OptionButton** llamado **Mayusculas** está activado. Observa que en este tipo de objeto miramos si está **activado** con un **True** y **desactivado** con un **False**. En el caso de estar activado lo que hacemos, en la primera parte del **If** es: **Etiqueta.Caption = UCase(Etiqueta.Caption)**. Esta instrucción funciona de la siguiente manera. Siempre que tenemos una igualdad la tenemos que leer de derecha a izquierda, así esta instrucción se leería de la siguiente forma. Cogemos el contenido de **Etiqueta**, cosa que hacemos con **Caption**, lo convertimos en mayúsculas con **UCase** y lo que tenemos (el contenido de la **Etiqueta** en mayúsculas) lo volvemos a poner en el **Caption** de nuestra **Etiqueta**.

Ahora tendríamos que mirar si lo que está activado es el **OptionButton** llamado **Minusculas**, pero no lo haremos mediante otro **If** ya que como vimos en la explicación de los objetos cuando seleccionamos uno dejamos de tener seleccionado el otro de tal forma que siempre tendremos uno seleccionado. Por lo que utilizaremos el **Else** del mismo **If** para controlar ya que si no tenemos seleccionado **Mayusculas** lo estará

**Minusculas**. Para poner el texto en minúsculas utilizaremos la instrucción **LCase**.

Con estas líneas comprobamos todas las posibles combinaciones que podemos hacer con nuestra pequeña aplicación. Intenta entender el pequeño código, si algo no lo entiendes tranquilo ya que más adelante explicaremos con más detenimiento estructuras e instrucciones.



10. Inicia una ejecución de prueba pulsando **F5** o pulsando el siguiente botón.

11. Realiza las pruebas que quieras sobre la aplicación.

Recuerda que solo se copiarán y se visualizarán los cambios cuando pulsemos el botón Copiar.

12. Finaliza la ejecución de la aplicación cerrando la pantalla.

## Ejercicio 6 – Guardar el formulario y el proyecto

Cuando realizamos una aplicación como la que hemos hecho en esta lección hemos creado una o varias ventanas llamadas **formularios** y al conjunto de estos **formularios** le llamamos **proyecto**.

Para grabar el **formulario** que hemos creado realizaremos los siguientes pasos.

1. Accede a **Guardar Form1 como...** dentro del menú **Archivo**.
2. Accede al directorio donde quieras guardar tus formularios, ponle el nombre que desees y pulsa en **Guardar**.

Fíjate que el formulario que has guardado tiene como extensión **frm**

Ahora guardaremos el **proyecto**.

3. Accede a **Guardar proyecto como...** dentro del menú **Archivo**.
4. Accede al mismo directorio donde has guardado tu **formulario**. Escribe **Primer programa** y pulsa en **Guardar**.

Fíjate que el proyecto se guardará con extensión **vbp**.

Ahora vamos a abrir un formulario nuevo, para así poder abrir el formulario recién guardado.

5. Escoge dentro del menú **Abrir** la opción **Nuevo Proyecto**.

Si te aparece una pantalla preguntando si deseas guardar los cambios responde negativamente.

6. En la siguiente pantalla pulsa en **Aceptar**.

Ahora ya tenemos nuestra primera aplicación guardada y en pantalla un nuevo **proyecto** para seguir trabajando. En lecciones futuras veremos como crear un archivo ejecutable de nuestra aplicación.

## Ejercicio 7 – Unidades de Temperatura

Se trata de realizar un programa sencillo que muestre la equivalencia entre las escalas de temperaturas en grados centígrados y grados Fahrenheit. En el centro del formulario aparece una barra de desplazamiento vertical que permite desplazarse con incrementos pequeños de 1º. C y grandes de 10º. C. Como es habitual, también puede cambiarse el valor arrastrando con el ratón el cursor de la barra. Los valores máximos y mínimos de la barra son 100º C y -100º C.

A ambos lados de la barra aparecen dos cuadros de texto donde aparecen los grados correspondientes a la barra en ambas escalas. Encima aparecen dos rótulos (labels) que indican la escala de temperaturas correspondiente. Completan la aplicación un botón SALIR que termina la ejecución y un menú FILE con la única opción SALIR, que termina asimismo la ejecución del programa.

La tabla a continuación indica los controles utilizados en este ejemplo junto con las propiedades y los valores correspondientes

CONTROL	PROPIEDAD	VALOR
<b>frmTemp</b>	Name	frmTemp
	Caption	Conversor de Temperaturas
<b>mnuFile</b>	Name	mnuFile
	Caption	&Archivo
<b>mnuFileSalir</b>	Name	mnuFileSalir
	Caption	&Salida
<b>cmdSalir</b>	Name	cmdSalir
	Caption	Salir
<b>txtCent</b>	Name	txtCent
	text	0
<b>txtFahr</b>	Name	txtFahr
	text	32
<b>vsbTemp</b>	Name	vsbTemp
	Min	100
	Max	-100
	SmallChange	1
	LargeChange	10
<b>lblCent</b>	Value	0
	Name	lblCent
<b>lblFahr</b>	Caption	Grados Centígrados
	Font	MS Sans Serif, 10
	Name	lblFahr
<b>lblFahr</b>	Caption	Grados Fahrenheit
	Font	MS Sans Serif, 10

Y el código del programa es el siguiente:

### Option Explicit

```
Private Sub cmdSalir_Click()
```

```
    Beep
```

```
    End
```

```
End Sub
```

```
Private Sub mnuFileExit_Click()
```

```
    End
```

```
End Sub
```

```
Private Sub vsbTemp_Change()
```

```
    txtCent.text = vsbTemp.value
```

```
    txtFahr.text = 32 + 1.8 * vsbTemp.value
```

```
End Sub
```

Conversor de Temperaturas

Archivo

Grados Centigrados

47

Grados Farenheit

Text2

Salir

Responde a lo siguiente:

PREGUNTA	RESPUESTA
¿Que son los controles en Visual Basic?	
¿Cuáles son los controles que usamos en este programa?	
¿Cómo ponemos nombre a nuestros controles?	
¿Cómo ponemos un valor de texto a nuestros controles?	
¿Cuál es la instrucción o comando que termina el programa?	
¿Dónde ponemos el código del programa?	

## Ejemplo 8 – Colores y posiciones

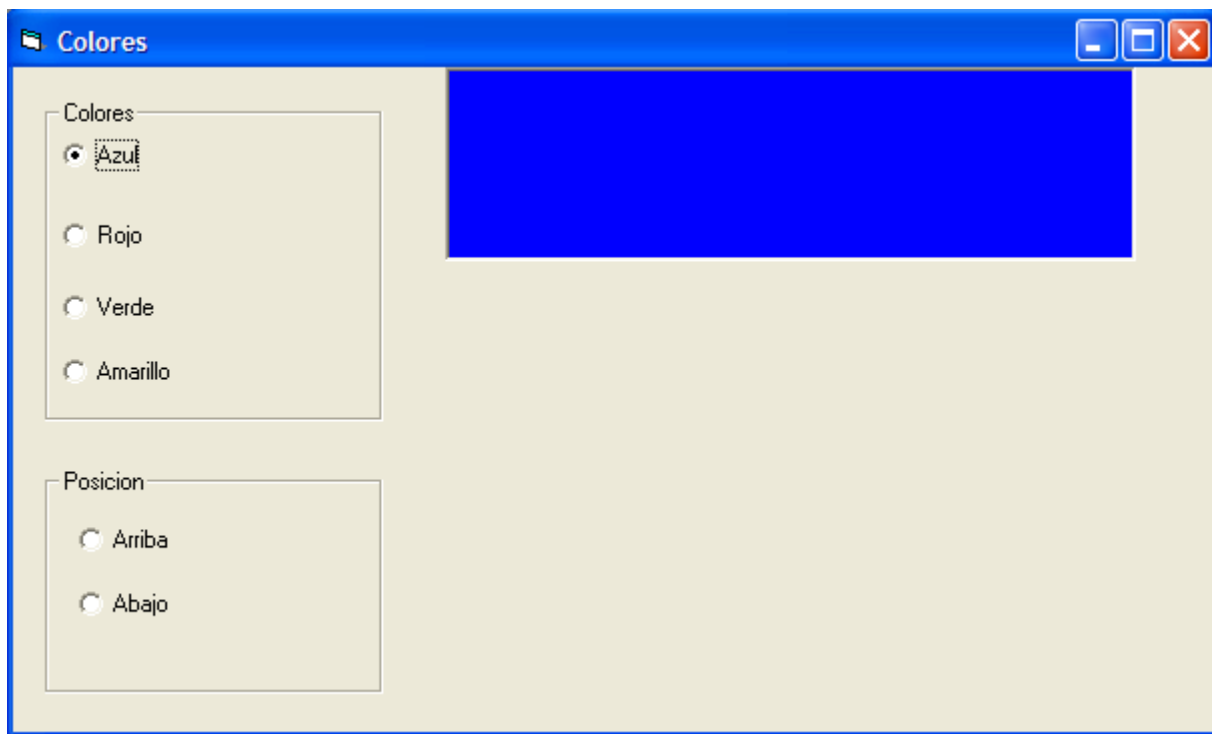
Ahora se presenta un sencillo ejemplo que permite mover una caja de texto por la pantalla, permitiendo a su vez representarla con cuatro colores diferentes.

Los archivos se llamarán Colores0.vbp y Colores0.frm

CONTROL	PROPIEDAD	VALOR
<b>frmColores0</b>	Name	frmColores0
	Caption	Colores
<b>fraColores</b>	Name	fraColor
	Caption	Colores
<b>optAzul</b>	Name	optAzul
	Caption	Azul
<b>optRojo</b>	Name	optRojo
	Caption	Rojo
<b>optAmarillo</b>	Name	optAmarillo
	Caption	Amarillo
<b>optVerde</b>	Name	optVerde
	Caption	Verde
<b>fraPosicion</b>	Name	fraPosicion
	Caption	Posicion
<b>optArriba</b>	Name	optArriba
	Caption	Arriba
<b>optAbajo</b>	Name	optAbajo
	Caption	Abajo
<b>txtCaja</b>	Name	txtCaja
	Text	""

Y el código es:

```
Option Explicit  
Private Sub Form_Load()  
    txtCaja.Top = 0  
End Sub  
  
Private Sub optArriba_Click()  
    txtCaja.Top = 0  
End Sub  
  
Private Sub optAbajo_Click()  
    txtCaja.Top = frmColores0.ScaleHeight - txtCaja.Height  
End Sub  
  
Private Sub optAzul_Click()  
    txtCaja.BackColor = vbBlue  
End Sub  
  
Private Sub optRojo_Click()  
    txtCaja.BackColor = vbRed  
End Sub  
  
Private Sub optVerde_Click()  
    txtCaja.BackColor = vbGreen  
End Sub  
  
Private Sub optAmarillo_Click()  
    txtCaja.BackColor = vbYellow  
End Sub
```



Responde a lo siguiente:

PREGUNTA	RESPUESTA
¿Cuáles son las propiedades de la caja de texto que estamos usando en este programa?	
¿Qué otros controles estamos viendo en este ejercicio?	
¿Cómo cambiamos el color de fondo de control de textbox?	

## Ejercicio 9 – MiniCalculadora

En este ejemplo se muestra una calculadora elemental que permite hacer las cuatro operaciones aritmeticas. Los archivos de este proyecto se pueden llamar minicalc.vbp y minicalc.frm

CONTROL	PROPIEDAD	VALOR
Form	Name	frmMinicalc
	Caption	Minicalculadora
textbox	Name	txtOper1
	Text	
textbox	Name	txtOper2
	Text	
textbox	Name	txtResult
	Text	
label	Name	lblOp
	Caption	
label	Name	lblEqual
	Caption	=
CommandButton	Name	cmdSuma
	Caption	+
CommandButton	Name	cmdResta
	Caption	-
CommandButton	Name	cmdMulti
	Caption	*
CommandButton	Name	cmdDiv
	Caption	/

Y a continuacion se muestra el codigo correspondiente a los procedimientos

### Option Explicit

```
Private Sub cmdDiv_Click()  
    txtResult.Text = val(txtOper1.Text) / val(txtOper2.text)  
    lblOp.Caption = "/"  
End Sub
```

```
Private Sub cmdProd_Click()  
    txtResult.Text = val(txtOper1.Text) * val(txtOper2.text)  
    lblOp.Caption = "*"   
End Sub
```

```
Private Sub cmdDiv_Click()  
    txtResult.Text = val(txtOper1.Text) / val(txtOper2.text)  
    lblOp.Caption = "/"  
End Sub
```

```
Private Sub cmdResta_Click()  
    txtResult.Text = val(txtOper1.Text) - val(txtOper2.text)
```

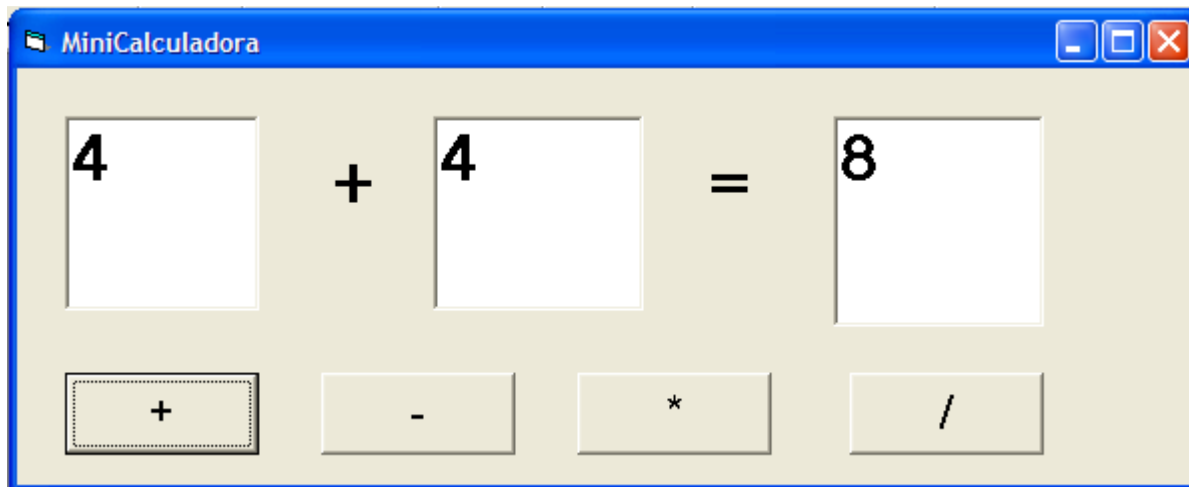


```

    lblOp.Caption = "-"
End Sub

Private Sub cmdSuma_Click()
    txtResult.Text = val(txtOper1.Text) + val(txtOper2.Text)
    lblOp.Caption = "+"
End Sub

```

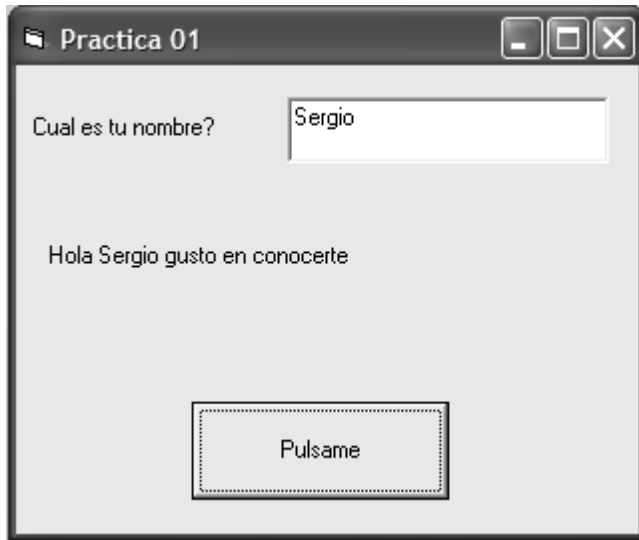


Responde a lo siguiente:

PREGUNTA	RESPUESTA
¿Para que usamos la funcion VAL() de VisualBasic?	
¿Qué operaciones estamos realizando?	
¿Dónde ponemos el codigo que realiza las operaciones?	

## PRACTICA 1- Preguntando tu nombre

Ahora, realiza tu primera practica con los dos controles basicos de Visual Basic: una caja de texto y un control de etiqueta en el cual te pregunte tu nombre y luego te salude. Quedaria asi:



## PRACTICA 2 – Preguntando tu nombre (Print)

Haz una variacion de la practica anterior, pero en lugar de usar el control de etiqueta, solamente usa el comando PRINT de Visual Basic. Para que te des una idea, quedaria mas o menos asi:

```
Dim saludo As String * 10
```

```
Private Sub Command1_Click()  
    Dim Nombre As String * 10  
    Nombre = "Joaquin"  
    Saludo = "Hola"  
    Print saludo; y; Nombre  
End Sub
```

## Ejercicio 10 – Calculadora sencilla

Vamos a familiarizarnos con algunas de las propiedades más importantes de los formularios, como puede ser la posición en la pantalla del formulario cuando se inicia la aplicación, el color de fondo, los botones de maximizar, minimizar y cerrar, etc.

### *Propiedades del formulario*

1. Inicia Visual Basic y haz lo necesario para que te aparezca un nuevo formulario en pantalla.

Una vez tenemos el formulario en pantalla vamos a cambiarle el tamaño. Recuerda que tienes varias maneras de hacerlo. Utiliza el sistema que tú prefieras. Mira la lección anterior.

2. Pon las propiedades Height a 5775 y Width a 6045.

Posición al iniciar la ejecución

3. Haz un clic en **Ventana** posición del formulario del menú Ver.



Observa como en algún lugar de la pantalla te ha aparecido una ventana como esta.

Esta ventana nos ofrece una simulación de lo que sería nuestro formulario dentro de la pantalla del ordenador.

4. Sitúate encima del recuadro blanco donde aparece la palabra **Form1**.

Observa como te ha aparecido un cursor, más o menos como este:



Si mantienes pulsado el botón izquierdo del ratón podrás ver como puedes mo- ver el

formulario a cualquier parte de la pantalla negra. Con esto conseguimos que el formulario en el momento de ejecutarse se inicie en el lugar que hemos situado el recuadro **Form1**.

5. Coloca el dibujo del formulario en una de las esquinas e inicia una ejecución de prueba. Acto seguido detén la ejecución de prueba.

Observa como el formulario aparece en el lugar de la pantalla que tu le has indicado. Con esta misma pequeña ventana podemos hacer que el formulario, siempre nos aparezca centrado en la pantalla.

6. Sitúate sobre el dibujo del formulario. Pulsa el botón derecho del ratón para que aparezca el menú contextual. Haz un clic en **Guías de resolución**.

Con esta opción podrás ver unas guías que te indican como sería la pantalla con resoluciones inferiores a la que tienes actualmente en tu ordenador.

7. Quita la opción **Guías de resolución** (pulsando otro clic en esta opción) y activa **Centro de la pantalla dentro de Posición inicial**.

Con esta otra opción lo que conseguirás es que el formulario siempre que se ejecute aparezca en el centro de la pantalla del usuario. En nuestra aplicación dejaremos activada esta opción.

Estas mismas opciones las podemos hacer desde la ventana de propiedades dentro de **StartUpPosition** con 4 opciones diferentes. **Manual**; **centrado** dentro de un formulario padre (está opción la explicaremos en futuras lecciones); **centrado en la pantalla** o **predefinido** por Windows (Esquina superior izquierda de la pantalla). Si te fijas son las mismas opciones que aparecen dentro del menú contextual al que hemos hecho referencia anteriormente.

Nosotros también podemos modificar la situación del formulario con respecto a los bordes interiores de la pantalla con las propiedades **Top** y **Left**. **Top** nos marca la distancia que existe entre la parte superior del monitor con la parte superior de nuestro formulario, mientras que **Left** nos marca la distancia entre la parte izquierda del monitor y la izquierda de nuestro formulario.

Si te molesta la ventana **Posición** del formulario la puedes cerrar.

### **Estilo del borde**

Con el estilo del borde, **BorderStyle**, lo que podemos conseguir es hacer, por ejemplo, que nuestra aplicación no tenga ningún tipo de borde, que no se pueda cambiar su tamaño, que el tamaño lo podamos variar como a nosotros nos apetezca,...

En nuestro caso nos interesa que no se pueda modificar el tamaño del formulario ya que al hacer más pequeño el formulario se podrían ocultar botones y no podríamos utilizar la aplicación correctamente. Lo que si permitiremos es que el usuario pueda minimizar la aplicación, pero no la pueda maximizar.

Dentro de **BorderStyle** tenemos 6 posibles opciones.

0 - None: Hace que en nuestra aplicación no aparezcan bordes.

1 - Fixed Single: Hace que el borde de la aplicación siempre quede fijo. Con esta opción podremos poner los botones minimizar o maximizar según nos convenga.

2 - Sizable: Esta opción es la que aparece por defecto al iniciar un nuevo formulario. Con esta opción podemos cambiar el tamaño del formulario a nuestro gusto.

3 - Fixed Double: Con esta opción podemos incluir el menú de control, la barra de título, pero no podemos incluir ni los botones maximizar ni minimizar. Esta ventana no podrá cambiarse de tamaño.

4 - Fixed Tool Window: Si activamos esta opción nos mostrará un formulario con la fuente del título reducida. No podremos modificar el tamaño del formulario. Este no aparecerá en la barra de tareas de Windows.

5 - Sizable Tool Window: Tendremos una ventana de tamaño ajustable. El tamaño de la fuente del título aparecerá reducida. El formulario no aparecerá en la barra de tareas.

Una cosa que hay que tener en cuenta es que estas opciones se ponen en funcionamiento en el momento que ejecutamos la aplicación. Otra cosa a tener en cuenta es que el menú de control que aparece sobre el icono de la aplicación también se modificará según las opciones de **BorderStyle** que hemos seleccionado y los botones de minimizar y maximizar que tengamos activados.

8. Coloca la propiedad **BorderStyle** de nuestro formulario a **1 - Fixed Single**.

Observa como los botones maximizar y minimizar han desaparecido de nuestro formulario, solo queda visible el botón cerrar.

9. Inicia una ejecución de prueba e intenta modificar el tamaño del formulario. Cuando termines detén la ejecución.

Vamos a colocar el botón **minimizar** para que el usuario pueda minimizar el formulario cuando le apetezca. Aunque esté esta opción activada el formulario seguirá sin “dejarse” cambiar el tamaño.

10. Sitúate sobre la propiedad **MinButton**.

Observa como esta propiedad tiene como valor **False**. Esto nos indica que el botón minimizar no está activado.

11. Haz doble clic sobre la palabra **MinButton** y observa como su valor cambia a **True**.

De esta manera hemos hecho que en nuestro formulario aparezca el botón minimizar. Observa como ha aparecido también el botón maximizar pero este no está activado. Para activarlo tendríamos que poner a **True** la propiedad **MaxButton**. En nuestro ejemplo no lo vamos a activar ya que no nos interesa que el usuario pueda maximizar nuestra aplicación.

Si queremos que el usuario no pueda mover por la pantalla la aplicación tendríamos que poner la propiedad **Moveable** a **False**. No es muy recomendado utilizar esta opción, excepto en casos muy específicos, ya que tenemos que dejar que el usuario pueda mover las aplicaciones por la pantalla para así poder visualizar el contenido de otras aplicaciones que están por detrás de esta.

### **Apariencia del formulario**

Vamos a cambiar el texto que aparece en el título del formulario. Recuerda como se hace según lo explicado en la primera lección.

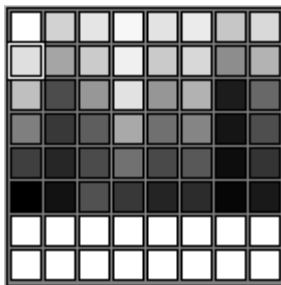
12. Escribe **Pequeña** calculadora como título de nuestra aplicación.

Ahora cambiaremos el icono que aparece en nuestra aplicación.

13. Coloca como icono de la aplicación **Misc18.ico** que se encuentra dentro del directorio **Graphics\Icons\Misc** dentro del directorio donde tengas instalado **Visual Basic**.

Ahora vamos a cambiar el color de fondo de nuestra **Pequeña calculadora**.

14. Accede a la propiedad **BackColor** y haz clic en la flecha para que se despliegue el menú de colores.



Observa que aparecen dos carpetas. Una llamada **Sistema**, donde aparecen los colores de todos los objetos que vienen determinados por Windows y **Paleta** donde nos aparecen diversos colores para elegir. La **Paleta** es parecida a la que mostramos en la derecha.

Si haces clic con el botón izquierdo sobre uno de los cuadrados blancos inferiores te aparecerá una pantalla, en la que podrás elegir un color entre todos los disponibles dentro de la paleta de colores de Windows.

15. Haz clic sobre el color que desees para el fondo de nuestra aplicación. Yo he seleccionado el gris claro. (Segunda fila, primera columna).

No pongas colores que cansen mucho a la vista ya que debemos pensar que nuestras aplicaciones pueden ser utilizadas por usuarios durante mucho rato con lo que le puede producir sensación de cansancio.

## ***Añadir objetos al formulario***

Vamos a situar en nuestro formulario los elementos que ya conocemos: **Label**, **TextBox** y **CommandButton**. Para ello os mostraremos una lista en la que aparecerá el **tipo** de elemento que deberéis añadir en nuestro formulario, el **texto** que debe aparecer, cual debe ser su **nombre** (en caso de necesitarlo), su **tamaño** y por último su **posición**.

Más adelante modificaremos la apariencia de los objetos que añadiremos ahora.

Repasa la primera lección cuando se explica como se añaden objetos nuevos, como se cambian de tamaño y como se sitúan en una posición determinada dentro del formulario.

16. Deberás añadir 6 objetos **Label**.

Será mejor que los vayas añadiendo y modificando de uno en uno.

Modifica las propiedades de cada **Label** para que queden de la siguiente forma:

### Label1

Caption: Calculadora

Left: 1920

Top: 0

### Label2

Caption: Primer operando

Left: 240

Top: 1320

### Label3

Caption: Segundo operando

Left: 2280

Top: 1320

Label4

Caption: Resultado

Left: 4680

Top: 1320

Label5

Caption: Lista de operaciones

Left: 480

Top: 3360

Label6

Caption: Operaciones con la lista

Left: 3360

Top: 4560

Label7

Caption: 0

(Nombre): MostrarResultado

Left: 4560

Top: 1560

BorderStyle: 1 - Fixed Single

Observa que a los 6 primeros **Label** que hemos añadido a nuestro formulario, no le hemos puesto (**Nombre**) a ninguno. Esto es debido a que durante la ejecución de esta aplicación estos objetos no deberán sufrir ningún tipo de modificación con lo que el nombre no nos interesa.

En cambio, el **Label7** será donde nos aparecerá el resultado de la operación que deseamos realizar.

17. Añade 2 objetos **CommandButton**.

Modifica las propiedades de cada **CommandButton** para que queden de la siguiente forma:

Command1

Caption: Borrar

(Nombre): Borrar

Left: 3000

Top: 2400

Width: 1215



Height: 495

Command2

Caption: Calcular

(Nombre): Calcular

Left: 4320

Top: 2400

Width: 1215

Height: 495

Recuerda como activar las teclas del modo abreviado de cada **Command**.

Ejemplo: Botón Calcular Alt+C.

18. Añade 2 objetos **TextBox**.

Modifica las propiedades de cada **TextBox** para que queden de la siguiente forma:

Text1

Text: (Borra el texto actual)

(Nombre):

PrimerOperando Left: 240

Top: 1560

Text2

Text: (Borra el texto actual)

(Nombre): SegundoOperando

Left: 2400

Top: 1560

Observa que en muchos de los objetos que hemos añadido hasta el momento no hemos especificado el tamaño, esto lo haremos más adelante cuando modifiquemos otras nuevas propiedades de estos objetos.

## ***Modificar propiedades de varios objetos simultáneamente***

Vamos a modificar una propiedad que tendrán en común varios objetos.

Si varios objetos que tenemos en nuestro formulario cumplen una misma propiedad podemos hacer dos cosas: podríamos ir seleccionando objeto a objeto y modificar la propiedad en cada uno de ellos o seleccionarlos todos y modificar de una sola vez la propiedad con lo que quedarían todos los objetos modificados.

19. Haz un clic sobre **Calculadora**.

20. Pulsa la tecla **Control** y mientras la mantienes pulsada haz clic en **Primer operando**, **Segundo operando**, **Resultado**, **Lista de operaciones** y **Operaciones con la lista**.

Observa como han quedado seleccionados todos los elementos que hemos marcado. Observa también como la lista de propiedades ha cambiado, solo se muestran las propiedades que podemos cambiar de forma conjunta a todos los objetos seleccionados.

Si te fijas en los objetos seleccionados podrás observar que tienen un fondo de color gris oscuro que delimita su tamaño. (Esto solo lo podrás ver si el color que escogiste para el formulario es diferente a este gris). Lo que vamos a hacer es ver una nueva propiedad que nos hará que estos objetos sean transparentes, de esta manera conseguiremos que solo se vea el texto y no el tamaño de dicho objeto.

21. Pulsa **F4** para acceder a las propiedades.

22. Haz un doble clic sobre la propiedad **BackColor** verás como todos los objetos seleccionados pasan de ser opacos a transparentes.

### ***Fuentes de letra en modo edición.***

La gran mayoría de los objetos que podemos añadir a un formulario contienen texto. Este texto también puede modificarse para hacer más vistosa o más clara nuestra aplicación. El formato de texto se puede cambiar desde el modo diseño o desde el modo de ejecución (como ya vimos en la lección anterior). En este capítulo explicaremos con más detenimiento ambos sistemas.

23. Si todavía mantienes seleccionados los objetos que hemos seleccionado en los anteriores puntos sólo debes hacer un clic, manteniendo pulsada la tecla **Control**, sobre **Calculadora**, para quitar la selección de este objeto. Si no mantienes la selección, vuelve a seleccionar los objetos que antes teníamos seleccionados pero esta vez sin el texto **Calculadora**.

Esto lo hemos hecho porque todos los objetos que están seleccionados tienen el mismo formato de letra mientras que el título **Calculadora** tiene otro formato.

24. Pulsa **F4**.

25. Accede a la propiedad **Font**.

Observa que esta propiedad está vacía. Esto siempre ocurre en el momento en el

que tenemos diferentes objetos seleccionados.

26. Pulsa un clic sobre el botón con tres puntos suspensivos que aparece en dicha propiedad.

Acto seguido aparece un cuadro de diálogo como este:

Dentro de la lista **Fuente** podremos seleccionar uno de los tipos de letras que tenemos instalado en nuestro ordenador. En el apartado **Estilo** de fuente podremos seleccionar entre cuatro opciones **Normal (ejemplo)**, **Cursiva (ejemplo)**, **Negrita (ejemplo)**, **Negrita cursiva (ejemplo)**. Juntamente con el estilo seleccionado podemos aplicar dos **Efectos** diferentes como es: **Tachado (ejemplo)** o **Subrayado (ejemplo)**. También podemos hacer una mezcla de los diferentes formatos de letra para así poder obtener algo así (**ejemplo**): negrita cursiva con subrayado y tachado. También podremos modificar el **Tamaño** de la fuente seleccionada. Debemos tener cuidado con esta propiedad ya que según el tamaño que seleccionemos podría ser que no se viera completamente el contenido de la información que deseamos mostrar.

27. Haz un clic en **Negrita**. Acepta la ventana.

Observa los cambios. Las demás opciones las dejaremos como están. Si no ves todo el contenido de estos elementos, no pasa nada.

28. Haz un clic en cualquier parte de la pantalla para quitar la selección.

29. Haz un clic sobre **Calculadora**.

30. Accede a la propiedad **Font**.

31. Accede al cuadro de diálogo **Fuente**.

32. Modifica el **tamaño** a **18** y haz que aparezca **Subrayado**.

Antes de aceptar la ventana observa el recuadro de **Ejemplo**. En este recuadro podrás ver una simulación de cómo quedarán las modificaciones que has hecho.

33. Acepta el cuadro de diálogo.

34. Selecciona el **TextBox** que lleva como (**Nombre**) **PrimerOperando** y **SegundoOperando**, junto con el **Label** llamado **MostrarResultado**.

35. Accede al cuadro de diálogo **Fuente**.

36. Cambia el **Tamaño** a **18**. Acepta el cuadro de diálogo.

## Modificar tamaños

Vamos a modificar el tamaño de estos 3 últimos objetos modificados.

37. Modifica el tamaño a 1215 x 540

## ***Fuentes de letra en modo ejecución***

Como ya vimos en la lección anterior los estilos de fuente se pueden modificar mientras estamos ejecutando el programa. Esto se consigue modificando las propiedades de **estilo** de fuente de alguno de los objetos insertados en nuestro formulario.

Vamos a imaginarnos que tenemos un objeto llamado **Texto** en nuestro formulario de trabajo al cual le modificaremos los estilos de fuente.

Para modificar un estilo de fuente como puede ser **negrita**, **cursiva**, **tachado** y **subrayado** utilizamos unas propiedades de tipo **booleano**<sup>1</sup>. Su sintaxis es exactamente igual que en el caso de cualquier otra propiedad. Deberemos escribir el nombre del objeto que queremos modificar seguido de un punto y una de estas cuatro propiedades: **FontBold** (**Negrita**), **FontItalic** (**Itálica**), **FontStrikethru** (**Tachado**) o **FontUnderline** (**Subrayado**), después el signo igual (=) y el valor **True** o **False** según nos interese activarlo o desactivarlo. (También podríamos poner **Font.Bold**, **Font.Italic**, **Font.Strikethru** o **Font.Underline**).

Por ejemplo, imaginemos que tenemos un **botón** que al pulsarlo queremos que el objeto **Texto** cambie a **negrita**. Dentro del objeto **botón** escribiremos la siguiente línea de código **Texto.FontBold = True** esto hará que el **Texto** aparezca en **negrita**. Si ya está en **negrita** no ocurrirá nada. Si queremos que aparezca el texto “normal” podríamos poner en otro **botón** la línea **Texto.FontBold = False**, de esta manera tendremos un **botón** que activa la negrita y otro que la desactiva. Esto funciona exactamente igual para cualquiera de las otras propiedades.

Si te fijas en este caso tenemos que diseñar dos botones para activar y desactivar la negrita, pero podemos hacer que un mismo botón haga las dos cosas, o cualquier otra propiedad, según la que exista en este momento. Lo explicaremos de otra forma; si el texto está en negrita se desactivará la negrita y si el texto no está en negrita se activará la negrita. Esto se consigue con la siguiente línea: **Texto.FontBold = Not Texto.FontBold**. La partícula **Not** hace que la propiedad se alterne, si está en **False** se convierte en lo contrario **True** y si su valor es **True** se convierte en **False**.

También podemos cambiar el tipo de fuente, esto lo haremos con la propiedad **FontName**. Esta propiedad no es de tipo **Booleana** ya que tenemos que especificar el

nombre de la fuente que queremos insertar. La sintaxis sería de la siguiente forma: **Nombre del objeto** seguido de un punto, la propiedad **FontName**, un igual y entre comillas dobles el nombre de la fuente. Por ejemplo: **Texto.FontName** = «Verdana».

Otra propiedad que tenemos para cambiar nuestro estilo de fuente es: **FontSize**, con esta propiedad lo que conseguimos es modificar el tamaño de la fuente. Esta propiedad tampoco es de tipo **booleana** ya que deberemos especificar el tamaño de la fuente. El tamaño se expresa en puntos. El tamaño máximo es de **2160** puntos. Los puntos son de tipo numérico con lo que la sintaxis sería de la siguiente manera: **Nombre del objeto** seguido de un punto, la propiedad **FontSize**, un igual y el número que indicará el tamaño de la fuente de letra. Por ejemplo: **Texto.FontSize** = 12.

Como practica adicional puedes crear un nuevo formulario para practicar estas nuevas propiedades.

## ***Tamaño automático***

Ahora vamos a modificar el tamaño de los cuadros de texto que tenemos en nuestro formulario.

Emplearemos otra nueva propiedad de estos objetos que es el ajuste automático del tamaño con respecto al texto que hay en su interior.

38. Selecciona todos los elementos de texto que tenemos hasta el momento, menos el que tiene como (**Nombre**) **MostrarResultado**.

39. Accede a las propiedades y cambia a **True** la propiedad **AutoSize**.

Observa como los puntos de selección de cada uno de los objetos se ha aproximado hasta el texto. Si nosotros ahora modificásemos la propiedad **Caption** veríamos como el tamaño del objeto cambia según el tamaño del texto que hay dentro de dicho objeto.

## ***Alineación del texto***

En nuestra práctica nosotros vamos a trabajar con diferentes números que iremos introduciendo en las casillas de primer y segundo operando para obtener un resultado.

Si nosotros utilizamos casillas de texto o etiquetas para que el usuario introduzca o visualice texto, normalmente se alinea a la izquierda (ya que es por donde comenzamos a escribir texto) y si trabajamos con números los alineamos a la derecha (para que todas las comas decimales en los números enteros estén juntas).

40. Selecciona solo **PrimerOperando** y accede a la propiedad **Text**.

41. Escribe la palabra **Texto**.

Esto lo hemos hecho para poder explicar mejor como actúa la alineación del texto en los diferentes objetos.

Observa como en este objeto al igual que en el **Label** **MostrarResultado** el texto está a la izquierda.

42. Accede a la propiedad **Alignment** de **MostrarResultado**.

Observa que tienes 3 posibles opciones. Esto lo podrás ver si despliegas la lista de esta propiedad. 0: izquierda, 1: derecha, 2: centro.

43. Selecciona la alineación a la derecha (1.- Right Justify).

Observa nuestro formulario y donde está alineado el texto de este objeto.

Vamos a hacer lo mismo con los objetos: **PrimerOperando** y **SegundoOperando**. Si quieres ver mejor los cambios y para asegurarte que lo haces correctamente puedes poner algo en la propiedad **Text** de **SegundoOperando**.

44. Selecciona **PrimerOperando** y **SegundoOperando** para trabajar con ambos objetos conjuntamente.

45. Accede a la propiedad **Alignment** y selecciona la opción correspondiente, para hacer que el texto de estos objetos aparezca alineado a la derecha.

Observa como en los dos objetos de tipo texto que tenemos seleccionados no ha ocurrido absolutamente nada. ¿A que es debido este comportamiento? Muy sencillo, si queremos que esta propiedad “funcione” tenemos que activar otra propiedad.

46. Accede a la propiedad **MultiLine** y ponla en **True**.

Observa que inmediatamente después de cambiar esta opción el texto pasa a estar alineado a la derecha.

La propiedad **MultiLine** lo que está haciendo es definir que en los dos objetos texto se puedan introducir varias líneas. Ten en cuenta que siempre que quieras una alineación a derecha o centro en objetos **Text** deberás activar la propiedad **MultiLine**.

47. Borra el contenido de los dos objetos seleccionados.

Observa que no podrás modificar el contenido de los objetos mientras estén los dos seleccionados.

Al acceder a la propiedad **Text** verás que hay la palabra (**Texto**) esto nos indica que **MultiLine** está activado y por lo tanto puede ser que dentro de este objeto puedan existir múltiples líneas de texto. Para eliminar lo que ya tenemos debemos pulsar en el botón con una flecha hacia abajo que aparece en esta propiedad y borrar el contenido.

## ***Delimitación de tamaño***

Ahora vamos a delimitar el tamaño de los números que podemos introducir en **PrimerOperando** y **SegundoOperando**. Esto lo conseguiremos con la propiedad **MaxLength**. Esta propiedad hará que no podamos introducir números con una cantidad de caracteres superiores a la que nosotros indiquemos. **Visual Basic** no nos dejará introducir más caracteres. No nos avisará de ninguna manera, simplemente no nos dejará introducir ningún carácter más.

48. Selecciona **PrimerOperando** y **SegundoOperando**.

49. Pulsa F4, para acceder a las propiedades.

50. Escribe 4 en **MaxLength**.

## ***Texto de ayuda***

Existe una propiedad, en la mayoría de los objetos que podemos añadir en nuestro formulario, que sirve para mostrar **ayuda** rápida al mantener el puntero del ratón durante unos segundos sobre el objeto deseado. Este texto suele ser corto y explícito dando una idea de para que sirve dicho control.

51. Selecciona **PrimerOperando**.

52. Accede a sus propiedades.

53. Sitúate sobre la propiedad: **ToolTipText**.

En esta propiedad podemos escribir lo que queremos que aparezca en el pequeño cuadro de ayuda al mantener el ratón durante unos segundos en el objeto seleccionado.

54. Escribe: **Introduce el primer operando**.

Realiza estas mismas operaciones con **SegundoOperando** y los botones **Calcular** y **Borrar**. Escribe el texto que creas conveniente, pensando que con el

botón **Calcular** se realizarán los cálculos pertinentes según la operación seleccionada (opciones que veremos en la siguiente lección) y el botón **Borrar** borra el contenido de **PrimerOperando**, **SegundoOperando** y **MostrarResultado**, para poder iniciar una nueva operación con diferentes operandos.

### ***OptionButton en modo gráfico***

Vamos a insertar unos controles que nos servirán para poder seleccionar cual de las cuatro operaciones (suma, resta, multiplicación o división) es la que deseamos realizar. Hemos escogido este elemento ya que solo podremos marcar uno de ellos a la vez.

En la primera lección ya utilizamos este tipo de objeto, pero aquí vamos a ver una nueva propiedad de este, ya que no trabajaremos con él con la apariencia que lo hicimos en la pasada lección, sino que tendrá apariencia de botón, pero con una imagen en su interior.

55. Inserta un **OptionButton**.

Observa como es su apariencia.

56. Ponle como **(Nombre)**: **Sumar**.

57. Accede a la propiedad **Style** y modifica su valor de **Standard** a **Graphical**.

Observa como su apariencia ahora es como un botón.

58. Borra el contenido de la propiedad **Caption**.

59. Accede a la propiedad **Picture** y selecciona **Misc18.ico** de **Graphics\Icons\Misc** dentro del directorio donde tengas instalado **Visual Basic**.

60. Cambia el tamaño a **540 x 540** y su posición a **1680, 600**.

61. Inserta 3 **OptionButton** más.

63. Modifica sus propiedades para que queden de la siguiente manera:

**Option1**

**Caption**: (Borra su contenido)

**(Nombre)**: **Restar**

**Posición**: **1680, 1200**

**Tamaño**: **540 x 540**

**Style**: **Graphical**

**Picture**: **Misc19.ico**



#### Option2

**Caption:** (Borra su contenido)

**(Nombre):** Multiplicar

**Posición:** 1680, 1800

**Tamaño:** 540 x 540

**Style:** Graphical

**Picture:** Misc20.ico

#### Option3

**Caption:** (Borra su contenido)

**(Nombre):** Dividir

**Posición:** 1680, 2400

**Tamaño:** 540 x 540

**Style:** Graphical

**Picture:** Misc21.ico

64. Asegúrate que la propiedad **Value** de objeto **Sumar** está en **True**.

65. Escribe en la propiedad **ToolTipText** de cada uno de estos objetos algo que le pueda servir de ayuda a los usuarios de esta aplicación, tal como vimos en puntos anteriores.

66. Realiza una ejecución de prueba. Selecciona las diferentes operaciones.

Observa que cuando se selecciona una, se quita la selección la que estaba seleccionada y así sucesivamente.

## **Ejercicio 11 – Declaracion de variables**

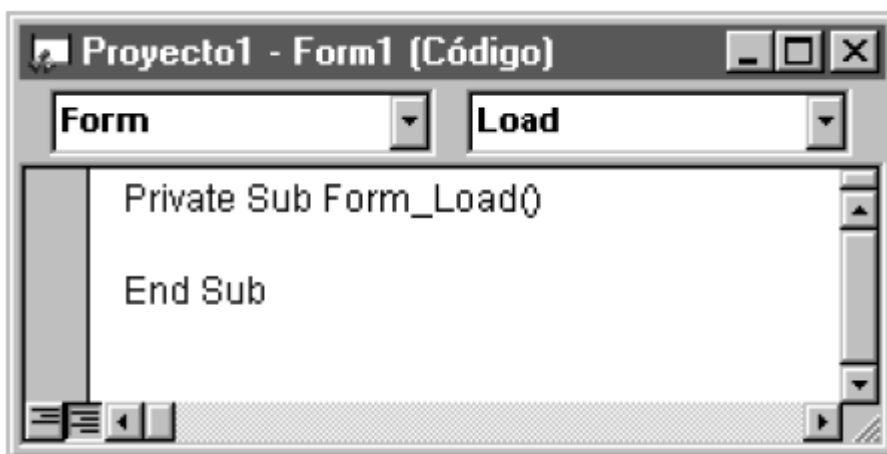
1. Abre un nuevo proyecto
2. Selecciona OPCIONES dentro del menu HERRAMIENTAS
3. De todas las carpetas selecciona EDITOR y activa la opcion REQUERIR DECLARACION DE VARIABLES
4. Acepta el cuadro de dialogo actual
5. Mira el codigo de este proyecto, con el menu VER – CODIGO o pulsa F7
6. Observa que no hay ninguna linea de codigo en nuestro proyecto
7. Cierra el proyecto actual, sin guardar los cambios
8. Abre un nuevo proyecto, con la opcion NUEVO PROYECTO de la opcion ABRIR. En el cuadro de dialogo que te aparece a continuacion deja la selección actual y pulsa en ACEPTAR
9. Mira el codigo del proyecto.

Responde a lo siguiente:

PREGUNTA	RESPUESTA
¿Qué es lo que cambio al modificar las opciones de Visual Basic?	

## Ejercicio 12 – Declaracion de variables explicitas

1. Con el ultimo proyecto en pantalla, quita la selección REQUERIR DECLARACION DE VARIABLES
2. Abre un proyecto que tengas grabado
3. Accede al codigo de cualquiera de los objetos que tienes en el formulario



4. Observa la pantalla con el codigo. Observa como en dicha ventana de codigo siempre aparece dos listas desplegables. La lista de la izquierda es donde se iran situando los **nombres** de los diferentes objetos que estan insertados en el formulario actual. Mientras que en la lista de la derecha apareceran los **eventos** del objeto que se este seleccionando en la lista de la izquierda.
5. Despliega la lista de la izquierda y selecciona la opcion **(General)** observa como la lista de la derecha cambia y aparece **(Declaraciones)**, si no aparece automaticamente despliega la lista y busca dicha opcion.
6. Cuando estas en este apartado ya puedes escribir **Option Explicit**.

Responde a lo siguiente:

PREGUNTA	RESPUESTA
¿Qué sucede si al utilizar una nueva variable y no esta declarada?	

## Ejercicio 13 – Variables

1. Crea un nuevo proyecto
2. Inserta dos **CommandButton** a los que llamaremos **Boton1** y **Boton2**.
3. Inserta un **Label** al que llamaremos **Valor**
4. Escribe dentro del **Boton1**, haciendo doble clic, estas lineas de codigo:

```
Private Sub Boton1_Click()  
    Dim Contador As Integer  
    Valor.Caption = Contador  
End Sub
```

5. Y dentro del **Boton2** estas otras:

```
Private Sub Boton2_Click()  
    Valor.Caption = Contador  
End Sub
```

Observa como en el primer boton hemos definido una variable llamada **Contador**, mientras que en el segundo boton no.

6. Realiza una ejecucion de prueba. Pulsa en el primer boton. Observa como el valor de la variable ha pasado a nuestro **Label**.
7. Pulsa ahora el segundo Boton.



Se produce un error, apareciendo una ventana como la que mostramos en esta imagen. Este error nos avisa que existe una variable que no esta definida. Aunque parezca que la tenemos definida no es así. La definicion de dicha variable esta en otro procedimiento.

8. Pulsa el boton Aceptar y observa donde se ha producido el error.
9. Deten la ejecucion de la aplicación.

Si deseas utilizar una variable con el mismo nombre en otro procedimiento deberas volverla a definir. Piensa que aunque se llamen exactamente igual, son variables diferentes ya que estan en procedimientos diferentes.

Si nosotros creamos las variables con **Dim** al volver a entrar dentro del evento donde se ha creado la variable, esta se vuelve a iniciar. Si queremos que dentro de un procedimiento el valor de una variable se conserve deberas definirla poniendo **Static** en lugar de **Dim**.

10. Modifica el codigo de nuestra aplicación para que quede de la siguiente forma:

---

```
Private Sub Boton1_Click()  
    Static Contador As Integer  
    Contador = Contador + 1  
    Valor.Caption = Contador  
End Sub
```

---

---

```
Private Sub Boton2_Click()  
    Dim Contador As Integer  
    Contador = Contador + 1  
    Valor.Caption = Contador  
End Sub
```

---

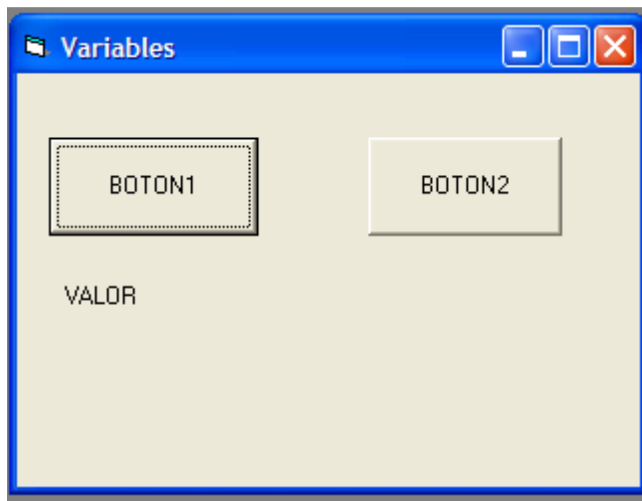
Lo que pretendemos con este ejemplo es que veas como utilizando una variable definida con **Static** se puede mantener el valor dentro de un procedimiento, mientras que la misma variable definida como **Dim** en otro procedimiento actual completamente diferente.

11. Realiza una ejecucion de prueba
12. Pulsa repetidamente el primer boton

Aunque cada vez que pulsamos el boton estamos entrando en el procedimiento la variable guarda el valor y se le siguen sumando 1 gracias a la linea **Contador = Contador+1**

13. Pulsa repetidamente el segundo boton  
 Observa como cada vez que se entra en este boton el valor vuelve a ser el mismo, ya que no se guarda el valor de las veces anteriores.

14. Vuelve a pulsar el primer boton  
El valor que teniamos almacenados en este procedimiento vuelve a surgir
15. Deten la ejecucion.  
Observa bien las diferencias entre estos dos tipos de asignacion de variables.



Responde a lo siguiente:

PREGUNTA	RESPUESTA
¿Cuál es la diferencia entre DIM y STATIC?	

## Ejercicio 14 – Declarando Constantes

Vamos a imaginas que queremos realizar una aplicación en la que partiendo de un numero inicial de alumnos, cada vez que pulsemos un boton el numero de alumnos aumente en 1.

1. Borra las lineas de codigo que hemos escrito en las practicas anteriores y escribe el siguiente codigo alli donde corresponda (Ten presente no estamos utilizando el segundo boton)

```
Option Explicit
Public Contador As Integer
Const Alumnos = 45
```

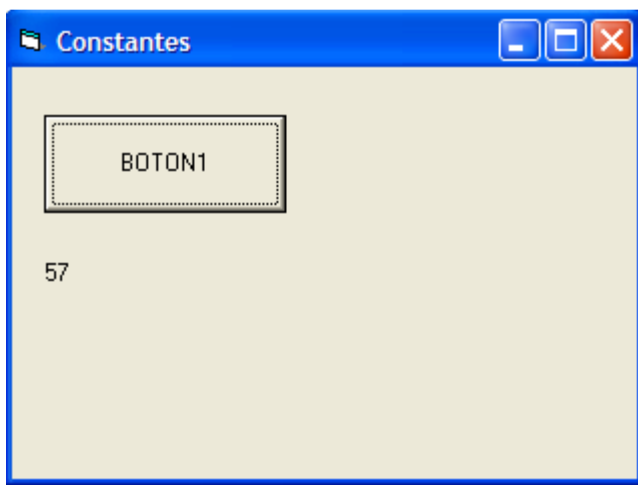
```
Private Sub Boton1_Click()  
    Contador = Contador + 1  
    Valor.Caption = Contador + Alumnos  
End Sub
```

2. Realiza una ejecucion de prueba
3. Finaliza dicha aplicación.

Podriamos pensar que no hace falta crear una constante llamada **Alumnos** donde introdujeramos el numero de alumnos que tenemos. Pero piensa que una constante es de suma utilidad en el momento que estamos realizando una gran aplicación en la que surge muchas veces una cantidad con la que tenemos que trabajar.

Ejemplo: imagina que tienes una aplicación con cientos de lineas en la que calculas el promedio de notas de la clase, el promedio de faltas en un trimestre, etc. Bien, pues en todos estos calculos necesitas saber el numero de alumnos que tienes. Si utilizaras esta misma aplicación otros años deberias cambiar el numero de alumnos. Entonces tendrias que buscar linea a linea alli donde realizas dichos calculos, para cambiar el numero de alumnos. En cambio, si utilizas una constante, con solo cambiar el valor de la constante, todos los cambios ya estan hechos.

4. Modifica el valor de la constante **Alumnos**.
5. Realiza otra ejecucion del programa.  
Observa que funciona exactamente igual, pero con valores diferentes.
6. Deten la ejecucion del programa.



Responde a lo siguiente:

PREGUNTA	RESPUESTA
¿Cuál es la diferencia entre una variable y una constante?	

## PRACTICA 3 – Convertir Horas

Realizar un programa que al darle la cantidad de horas las convierta a minutos y segundos.

The screenshot shows a Windows application window titled "Form1". The window has a blue title bar with standard minimize, maximize, and close buttons. The main area has a light beige background. At the top left, the text "Numero de Horas" is displayed in a bold, italicized font. To its right is a white text input box. Below this input box is a button labeled "Convertir" with a dashed border. At the bottom left, the label "Minutos" is followed by a white text input box. To the right of this, the label "Segundos" is followed by another white text input box. At the bottom right of the window is a button labeled "Salir".

## PRACTICA 4 – Calculo de Areas

Realice ahora el calculo del area de estos poligonos: CUADRADO, RECTANGULO y TRIANGULO para que practique los operadores y variables, asi como el manejo de los controles.

- a) Area del Cuadrado: Lado x Lado
- b) Area del Rectangulo: Lado menor x Lado mayor
- c) Area del Triangulo: (Base X Altura ) / 2

El programa debe quedar mas o menos asi:

The screenshot shows a Visual Basic application window titled "Calcular Areas". The window has a blue title bar with standard Windows controls (minimize, maximize, close). The main area has a light beige background and contains the text "Calcular areas de diferentes poligonos". Below this, there are three distinct input sections, each enclosed in a rounded rectangle:

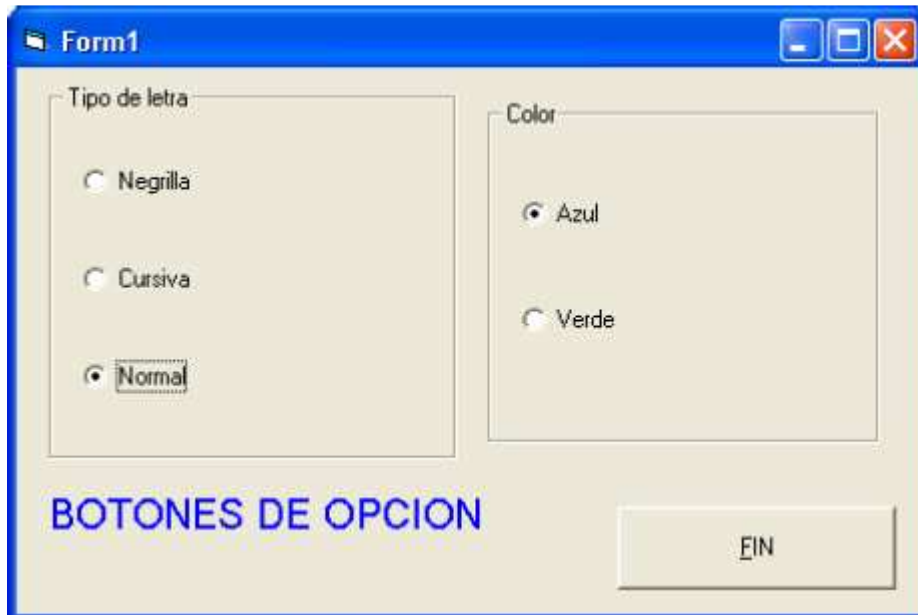
- Section 1 (Square):** Labeled "Valor del lado:" followed by a text box containing "0". To the right is a button labeled "Cuadrado".
- Section 2 (Rectangle):** Labeled "Valor del lado mayor:" followed by a text box containing "0", and "Valor del lado menor:" followed by a text box containing "0". To the right is a pink button labeled "Rectangulo".
- Section 3 (Triangle):** Labeled "Valor de la base:" followed by a text box containing "0", and "Valor de la altura:" followed by a text box containing "0". To the right is a yellow button labeled "Triangulo".

At the bottom of the window, the word "RESULTADO" is displayed in large, bold, black capital letters.



## PRACTICA 5 – Botones de opcion y colores 2

Realiza un programa en el cual puedas cambiar las propiedades de una etiqueta, seleccionando de un conjunto de opciones el tipo de letra y el color de la misma.



Así, como vemos en la imagen, al seleccionar el tipo de letra normal y el color azul se mostrara en la etiqueta.

Para el color del texto de la etiqueta puedes usar la propiedad ForeColor, y los colores utiliza la notacion hexadecimal en lugar de las constantes de VisualBasic.

Azul = &HFF0000

Verde = &HFF00&

## Actividad de Aprendizaje 5 – Funciones Matematicas (L)

En este ejercicio veremos algunas funciones como la búsqueda de números al azar, raíz de números y redondeo de decimales. Usamos botones para ejecutar las funciones, cajas de texto para ingresar valores y labels para dar salida a los resultados.

The screenshot shows a Windows application window titled "Funciones". It contains four distinct sections, each with a button and a result label. The first section has a button "Número al Azar entre 1 y 10" and a result of 7. The second section has a button "Número al Azar entre Rangos", two input boxes for "Nº Inferior:" (10) and "Nº Superior:" (20), and a result of 12. The third section has a button "Ejecutar:", two input boxes for "Raíz:" (2) and "Número:" (81), and a result of 9. The fourth section has a button "Ejecutar:", two input boxes for "Número:" (4.45678) and "Decimales:" (2), and a result of 4.46.

El código es:

```
Option Explicit
Private Sub Command1_Click()
    'boton de número al azar
    Dim azar As Integer
    Randomize
    azar = Int(10 * Rnd) + 1
    Label3.Caption = azar
End Sub

Private Sub Command2_Click()
    'número al azar entre dos valores
    Dim azar As Integer
    Dim rangomenor As Integer
    Dim rangomayor As Integer
    Randomize
    rangomayor = CInt(Text2)
    rangomenor = CInt(Text1)
    azar = Int((rangomayor - rangomenor + 1) * Rnd + rangomenor)
    Label4 = azar
```

```

End Sub

Private Sub Command3_Click()' raíz de un número
    Dim numero As Integer
    Dim raiz As Integer
    raiz = CInt(Text3)
    numero = CInt(Text4)
    Label5.Caption = numero ^ (1 / raiz)
End Sub

Private Sub Command4_Click()'redondeo de un decimal
    Dim numero
    Dim decimales As Integer
    numero = Text6
    decimales = CInt(Text5)
    Label10.Caption = Round(numero, decimales)
End Sub

Private Sub Form_Activate()
    Text1.SetFocus
End Sub

```

## Ejercicio 15 – Calculo de Salario Neto (Formulas)

Elaborar una aplicación que permita calcular el salario neto de un trabajador en función del número de horas trabajadas, pago por hora de trabajo y un descuento fijo al sueldo bruto del 20 por 100. Los cálculos a efectuar para obtener el salario neto de un trabajador se muestran a continuación:

$$\text{SalarioBruto} = \text{HorasTrabajadas} * \text{PagoPorHora}$$

$$\text{Descuento} = 0.2 * \text{SalarioBruto}$$

$$\text{SalarioNeto} = \text{SalarioBruto} - \text{Descuento}$$

El diseño de la interfaz debe ser similar a la figura mostrada:

Para el desarrollo de esta aplicación, proceda a ubicar los siguientes controles en el formulario:

2 marcos

6 etiquetas

6 cajas de texto

3 botones de comando

Una vez ubicados los controles, establezca las propiedades según se indica:

#### Form1

Nombre	FrmSalarioNeto
BorderStyle	3-Fixed Dialog
Caption	Cálculo del salario neto
Moveable	False
StartPosition	2-CenterScreen

#### Frame1

Nombre	FraIngreso
Caption	Ingreso de datos:

**Frame2**

Nombre	FraSalida
Caption	Salida de datos:

**Label1**

Nombre	LblApellidosNombres
AutoSize	True
Caption	Apellidos y nombres:

**Label2**

Nombre	LblHorasTrabajadas
AutoSize	True
Caption	Horas trabajadas:

**Label3**

Nombre	LblPagoPorHora
AutoSize	True
Caption	Pago por hora S/.

**Label4**

Nombre	LblSalarioBruto
AutoSize	True
Caption	Salario bruto S/.

**Label5**

Nombre	LblDescuento
AutoSize	True
Caption	Descuento S/.

**Label6**

Nombre	LblSalarioNeto
AutoSize	True
Caption	Salario neto S/.

**Text1**

Nombre	TxtApellidosNombres
Text	

**Text2**

Nombre	TxtHorasTrabajadas
Text	

**Text3**

Nombre	TxtPagoPorHora
Text	

**Text4**

Nombre	TxtSalarioBruto
Text	

**Text5**

Nombre	TxtDescuento
Text	

**Text6**

Nombre	TxtSalarioNeto
Text	

### Command1

Nombre	CmdAceptar
Caption	&Aceptar

### Command2

Nombre	CmdLimpiar
Caption	&Limpiar

### Command3

Nombre	CmdSalir
Caption	&Salir

Luego de establecer las propiedades para los controles, debe añadir código a la aplicación. Para ello haga doble click sobre el botón Aceptar e ingrese el siguiente código:

```
Private Sub CmdAceptar_Click()  
    Dim HorasTrabajadas As Integer, PagoPorHora As Double  
    Dim SalarioBruto As Double  
    Dim Descuento As Double  
    Dim SalarioNeto As Double  
    HorasTrabajadas = Val(TxtHorasTrabajadas)  
    PagoPorHora = Val(TxtPagoPorHora)  
    SalarioBruto = HorasTrabajadas * PagoPorHora  
    Descuento = 0.2 * SalarioBruto  
    SalarioNeto = SalarioBruto - Descuento  
    TxtSalarioBruto = Str(SalarioBruto)  
    TxtDescuento = Str(Descuento)  
    TxtSalarioNeto = Str(SalarioNeto)  
End Sub
```

A continuación haga doble click sobre el botón Salir y añada el siguiente código:

```
Private Sub CmdSalir_Click()
```

```
End
```

```
End Sub
```

Guarde y luego ejecute la aplicación que acaba de crear. Pruebe ingresando diferentes valores.

Luego, haga doble click sobre el botón Limpiar y añada el siguiente código:

```
Private Sub CmdLimpiar_Click()
```

```
TxtApellidosNombres = ""
```

```
TxtHorasTrabajadas = ""
```

```
TxtPagoPorHora = ""
```

```
TxtSalarioBruto = ""
```

```
TxtDescuento = ""
```

```
TxtSalarioNeto = ""
```

```
TxtApellidosNombres.SetFocus
```

```
End Sub
```

Guarde y ejecute su aplicación. ¿Cómo afecta el código añadido en el botón Limpiar a la aplicación?. Anote sus observaciones.



## Ejercicio 16 – Restaurante (Formulas)

El menú de un restaurante rápido se muestra a continuación:

MENÚ	PRECIO (S/.)
Hamburguesa	2.50
Cerveza	4.00
Gaseosa	3.00
Ensalada	1.50
Salchichas	2.00
Refresco	1.00
Sopa	1.50
Postre	1.50

Se desea construir una aplicación que calcule las ventas totales al final del día, así como los impuestos a pagar (18 por 100).

La interfaz de entrada y salida deberá ser similar a la figura mostrada a continuación:

The screenshot shows a Windows-style application window titled "Restaurante". On the left, there are labels for menu items followed by input boxes for their quantities: Hamburguesa: 43, Cerveza: 72, Gaseosa: 28, Ensalada: 15, Salchichas: 39, Refresco: 34, Sopa: 17, and Postre: 22. In the center, there is a menu table with two columns: "Menú" and "Precio S/.". The table lists the same items as the menu on the left with their respective prices. On the right side of the window, there are three buttons: "Aceptar" (highlighted with a dotted border), "Limpiar", and "Salir" (which has a hand cursor icon over it). At the bottom of the window, there are two more input fields: "Venta total S/." with the value 672.50 and "Impuesto S/." with the value 121.05.

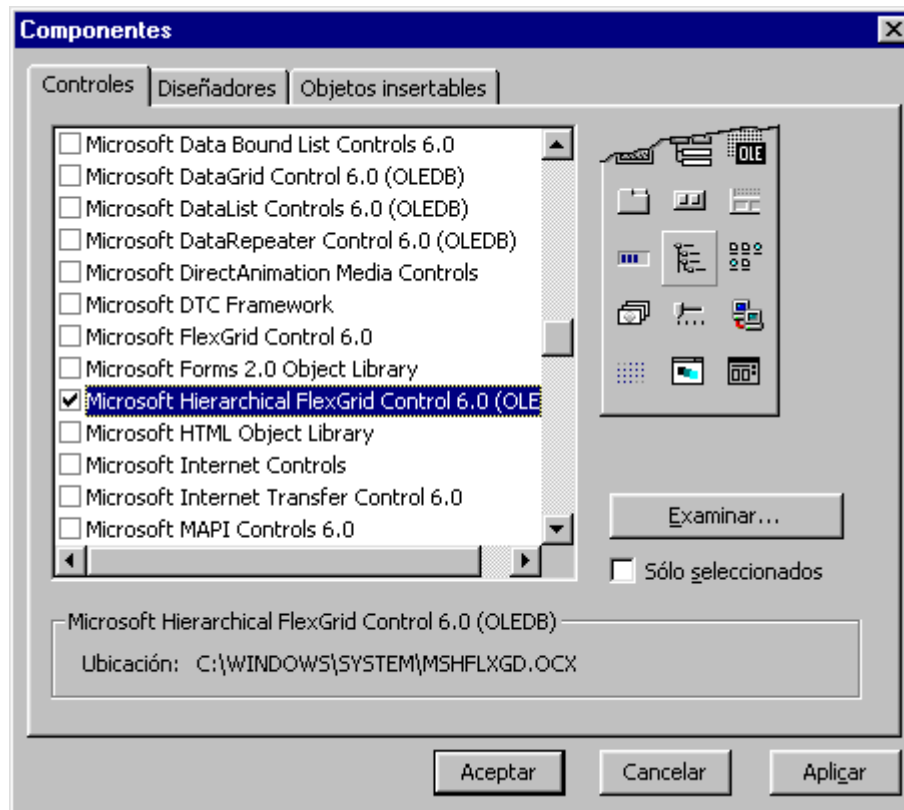
Menú	Precio S/.
Hamburguesa	2.50
Cerveza	4.00
Gaseosa	3.00
Ensalada	1.50
Salchichas	2.00
Refresco	1.00
Sopa	1.50
Postre	1.50

Hamburguesa: 43  
Cerveza: 72  
Gaseosa: 28  
Ensalada: 15  
Salchichas: 39  
Refresco: 34  
Sopa: 17  
Postre: 22

Venta total S/. 672.50  
Impuesto S/. 121.05

Buttons: Aceptar, Limpiar, Salir

Para el diseño del cuadro de menú utilizaremos el control Microsoft Hierarchical FlexGrid. Para tal fin, seleccione el Menú Proyecto y elija la opción Componentes:



A continuación active la casilla de verificación Microsoft Hierarchical FlexGrid Control 6.0 OLEDB y haga click sobre el botón Aceptar. En seguida este control se añadirá al Cuadro de Herramientas.

Para el desarrollo de esta aplicación, proceda a ubicar los siguientes controles en el formulario:

10 etiquetas

10 cajas de texto

1 control MSHFlexGrid

3 botones de comando

En seguida, elabore el diseño de entrada y salida. Para ello proceda a establecer las propiedades según se indica a continuación:

#### Form1

Nombre	FrmRestaurante
BorderStyle	3-Fixed Dialog

Caption	Restaurante
Moveable	False
StartPosition	2-CenterScreen

#### Label1

Nombre	LblHamburguesa
AutoSize	True
Caption	Hamburguesa:

#### Label2

Nombre	LblCerveza
AutoSize	True
Caption	Cerveza:

#### Label3

Nombre	LblGaseosa
AutoSize	True
Caption	Gaseosa:

#### Label4

Nombre	LblEnsalada
AutoSize	True
Caption	Ensalada:

#### Label5

Nombre	LblSalchichas
AutoSize	True
Caption	Salchichas:

#### Label6

Nombre	LblRefresco
AutoSize	True
Caption	Refresco:

#### Label7

Nombre	LblSopa
AutoSize	True
Caption	Sopa:

#### Label8

Nombre	LblPostre
AutoSize	True
Caption	Postre:

#### Label9

Nombre	LblVentaTotal
Caption	Venta total S/.

#### Label10

Nombre	LblImpuesto
Caption	Impuesto S/.

#### MSHFlexGrid1

Nombre	GrdMenu
Font	Arial (Negrita 10)
FontFixed	Arial (Negrita 10)

#### Text1

Nombre	TxtHamburguesa
Text	

**Text2**

Nombre	TxtCerveza
Text	

**Text3**

Nombre	TxtGaseosa
Text	

**Text4**

Nombre	TxtEnsalada
Text	

**Text5**

Nombre	TxtSalchichas
Text	

**Text6**

Nombre	TxtRefresco
Text	

**Text7**

Nombre	TxtSopa
Text	

**Text8**

Nombre	TxtPostre
Text	

**Text9**

Nombre	TxtVentaTotal
Locked	True
Text	

**Text10**

Nombre	TxtImpuesto
Locked	True
Text	

**Command1**

Nombre	CmdAceptar
Caption	&Aceptar

**Command2**

Nombre	CmdLimpiar
Caption	&Limpiar

**Command3**

Nombre	CmdSalir
Caption	&Salir
Picture	C:\Archivos de programa\Microsoft Visual Studio\Common\Graphics\Icons\Arrows\Point04.ico
Style	1-Graphical

En primer lugar debemos cargar los datos a la cuadrícula. Esto lo vamos a realizar en tiempo de ejecución al momento de cargarse en memoria el formulario. Para ello, haga doble click sobre el formulario y añada el siguiente código:

```

Private Sub Form_Load()
    GrdMenu.Cols = 2
    GrdMenu.Rows = 9
    GrdMenu.FixedCols = 0
    GrdMenu.FixedRows = 1
    GrdMenu.TextArray(0) = "Menú"
    GrdMenu.TextArray(1) = "Precio"
    GrdMenu.TextArray(2) = "Hamburguesa"
    GrdMenu.TextArray(3) = "2.50"
    GrdMenu.TextArray(4) = "Cerveza"
    GrdMenu.TextArray(5) = "4.00"
    GrdMenu.TextArray(6) = "Gaseosa"
    GrdMenu.TextArray(7) = "3.00"
    GrdMenu.TextArray(8) = "Ensalada"
    GrdMenu.TextArray(9) = "1.50"
    GrdMenu.TextArray(10) = "Salchichas"
    GrdMenu.TextArray(11) = "2.00"
    GrdMenu.TextArray(12) = "Refresco"
    GrdMenu.TextArray(13) = "1.00"
    GrdMenu.TextArray(14) = "Sopa"
    GrdMenu.TextArray(15) = "1.50"
    GrdMenu.TextArray(16) = "Postre"
    GrdMenu.TextArray(17) = "1.50"
End Sub

```

Luego debemos añadir el código que se encargará de realizar los cálculos. Para tal fin haga doble click sobre el botón Aceptar y proceda a ingresar lo siguiente:

```

Private Sub CmdAceptar_Click()
    Dim Hamburguesa As Integer, Cerveza As Integer
    Dim Gaseosa As Integer, Ensalada As Integer
    Dim Salchichas As Integer, Refresco As Integer
    Dim Sopa As Integer, Postre As Integer
    Dim VentaTotal As Double, Impuesto As Double
    Hamburguesa = Val(TxtHamburguesa)
    Cerveza = Val(TxtCerveza)
    Gaseosa = Val(TxtGaseosa)

```

```

Ensalada = Val(TxtEnsalada)
Salchichas = Val(TxtSalchichas)
Refresco = Val(TxtRefresco)
Sopa = Val(TxtSopa)
Postre = Val(TxtPostre)
VentaTotal = Hamburguesa * 2.5 + Cerveza * 4.0 _
          + Gaseosa * 3.0 + Ensalada * 1.5 + Salchichas * 2.0 _
          + Refresco * 1.0 + Sopa * 1.5 + Postre * 1.5
Impuesto = 0.18 * VentaTotal
TxtVentaTotal = Str(VentaTotal)
TxtImpuesto = Str(Impuesto)
End Sub

```

Luego, haga doble click sobre el botón Limpiar y añada el siguiente código:

```

Private Sub CmdLimpiar_Click()
    TxtHamburguesa = "" : TxtCerveza = ""
    TxtGaseosa = "" : TxtEnsalada = ""
    TxtSalchichas = "" : TxtRefresco = ""
    TxtSopa = "" : TxtPostre = ""
    TxtVentaTotal = "" : TxtImpuesto = ""
    TxtHamburguesa.SetFocus
End Sub

```

A continuación haga doble click sobre el botón Salir y añada el siguiente código:

```

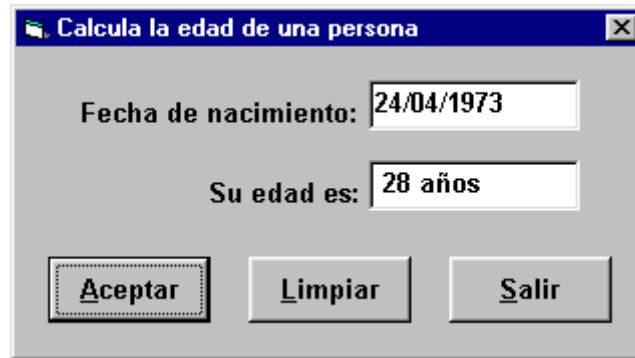
Private Sub CmdSalir_Click()
    End
End Sub

```



## Ejercicio 17 – Calculo de Edad

Elaborar una aplicación que permita calcular la edad de una persona a partir de su fecha de nacimiento. El diseño de la interfaz debe ser similar a la figura mostrada:



Para el desarrollo de esta aplicación, proceda a ubicar los siguientes controles en el formulario:

2 etiquetas

2 cajas de texto

3 botones de comando

Luego, proceda a establecer las propiedades según se indica a continuación:

### Form1

Nombre	FrmEdad
BorderStyle	3-Fixed Dialog
Caption	Calcula la edad de una persona

### Label1

Nombre	LblFecNac
AutoSize	True
Caption	Fecha de nacimiento:

**Label2**

Nombre	LblEdad
AutoSize	True
Caption	Su edad es:

**Text1**

Nombre	TxtFecNac
Text	

**Text2**

Nombre	TxtEdad
Locked	True
Text	

**Command1**

Nombre	CmdAceptar
Caption	&Aceptar
Default	True

**Command2**

Nombre	CmdLimpiar
Caption	&Limpiar

**Command3**

Nombre	CmdSalir
Cancel	True
Caption	&Salir

En seguida proceda a ingresar el siguiente código:

```

Private Sub CmdAceptar_Click()
    Dim FecNac As Date, Edad As Integer
    FecNac = CDate(TxtFecNac)
    Edad = CInt((Date - FecNac) / 365)
    TxtEdad = Str(Edad) & " años"
End Sub

```

Se deja como ejercicio para el estudiante el código asociado con los botones Limpiar y Salir, respectivamente.

## PRACTICA 6 – Cajero Automatico – Desglose de billetes

Se tiene un cajero automático el cual permite manipular cierta cantidad de dinero para lo cual dispone de los siguientes billetes: S/. 10.00, S/. 20.00, S/. 50.00, S/. 100.00 y S/. 200.00. Elaborar una aplicación que permita la lectura de la cantidad a retirar e indique el menor número de billetes a utilizar.

Denominación	Cantidad
Cantidad a retirar:	450.00
Billetes de S/. 10	0
Billetes de S/. 20	0
Billetes de S/. 50	1
Billetes de S/. 100	0
Billetes de S/. 200	2

Creo que a estas alturas Ud. ya entendió como funciona esto. Por ello, esta aplicación se deja como ejercicio para el estudiante.

## Ejercicio 18 – IF...Then..Else

Vamos a realizar una pequeña aplicación para que el ordenador nos diga, después de introducir la edad de una persona si es mayor de edad o no. Consideraremos la mayoría de edad a los 18 años.

En esta ejercicio simplificaremos los controles a utilizar, si lo deseas puedes ampliar el ejercicio tanto como desees.

1. Coloca en el lugar que desees de un formulario nuevo, un TEXTBOX y un LABEL  
El TEXTBOX lo utilizaremos para introducir la edad y el LABEL para que la computadora nos devuelva la cadena “Mayor de edad” en caso de ser mas grande o igual a 18 años, o la cadena “Menor de edad” en caso de ser menor de 18 años.
2. Cambia la propiedad (Nombre) del TEXTBOX y escribe: Edad
3. Cambia la propiedad (Nombre) del LABEL y escribe: Comentario.  
Puedes borrar el contenido de ambos objetos y modificar el aspecto como tu quieras.
4. Inserta un boton, en el cual colocaremos el codigo que se ejecutara al hacer clic sobre el.
5. Pon en la propiedad CAPTION de dicho boton Calcular. (No hace falta que cambies la propiedad (Nombre)).
6. Haz doble clic en el boton Calcular
7. Escribe el siguiente codigo:

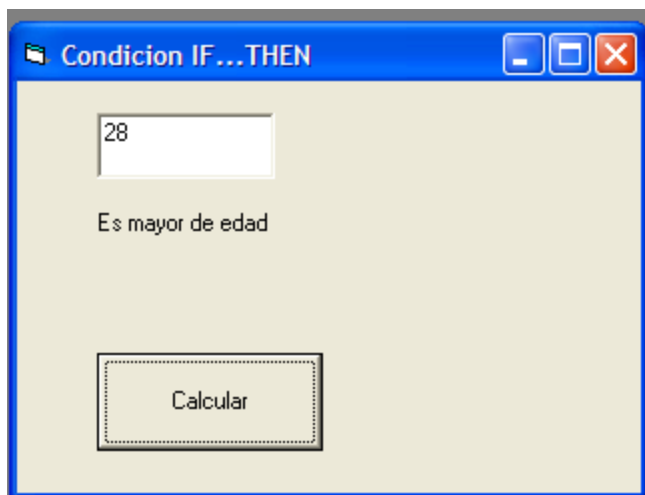
---

```
If Val(Edad.Text) < 18 Then  
Comentario.Caption = “Es menor de edad”  
Else  
Comentario.Caption = “Es mayor de edad”  
End If
```

---

Vamos a comentar el codigo anterior. La computadora lo primero que hace es mirar el valor del contenido del objeto llamado EDAD. Este objeto es de tipo texto y nosotros lo que estamos haciendo es mirar si es mayor o menor que un numero. Por esta razon nosotros convertimos el texto en valor numerico utilizando la orden VAL. La computadora se hace la pregunta “¿El contenido de EDAD es menor que 18?” Si la respuesta es **verdadera**, pasa a la primera parte del IF y escribe en el objeto COMENTARIO la frase “Es menor de edad”, si la respuesta es **falsa**, pasa a la segunda parte del IF, donde se escribe “Es mayor de edad”.

8. Haz una ejecucion de prueba.



## Ejercicio 19 – If..Then

Vamos a utilizar este mismo ejemplo, para explicar como escribiríamos el código en caso que solo quisiésemos que el ordenador nos devolviese un mensaje en caso que la edad fuera menor de 18 años.

1. Accede al código del botón Calcular.
2. Borra el código que hemos escrito anteriormente y escribe el siguiente código:

---

**If Val(Edad.Text) < 18 Then Comentario.Caption = "Es menor de edad"**

---

Todo este código deberá estar escrito en la misma línea.

Observa que no hemos utilizado la parte Else de la estructura, ni instrucciones que van en la parte de falso, ni End If indicando el final del código. Esto es debido a que solo queremos realizar una operación en caso de que sea Verdadero, con lo que no hace falta .cerrar. el If, y no ponemos Else porque no queremos instrucciones en el caso que sea la respuesta falsa.

3. Realiza un ejecución de prueba.

Ten en cuenta que si cuando ejecutas el programa escribes 21, la aplicación nos devolverá un mensaje diciendo .Es menor de edad.. Si acto seguido borras la edad y escribes 10, al pulsar el botón continuarás viendo en el cuadro de mensaje .Es menor de edad.. Esto no quiere decir que la aplicación funcione mal, ya que no existe ninguna instrucción que haga que en el caso de no ser mayor de edad se borre el mensaje.

Podemos decir que la aplicación funciona correctamente, pero está mal diseñada, ya que nos proporciona información incorrecta.

## Ejercicio 20 – If... Anidados

Imagina que lo que queremos ahora es que el ordenador nos devuelva, mirando la edad que introducimos, si es menor de 10 años, si tiene entre 10 y 20 años, si tiene entre 20 y 30 años o si es mayor de 30. Hasta este momento solo hemos visto la instrucción If para poder controlar un caso como el que planteamos.

Veamos que código tendríamos que usar para que funcione el caso anteriormente planteado.

1. Escribe estás líneas dentro del botón Calcular. Borra las instrucciones escritas anteriormente.

---

```
1 If Val(Edad.Text) < 10 Then  
2     Comentario.Caption = "Menos de 10 años"  
3 Else  
  
4     If Val(Edad.Text) < 20 Then  
5         Comentario.Caption = "Entre 10 y 20 años"  
6     Else  
7         If Val(Edad.Text) < 30 Then  
8             Comentario.Caption = "Entre 20 y 30  
años"  
9         Else  
10             Comentario.Caption = "Más de 30  
años"  
11             End If  
12         End If  
13     End If
```

---

2. Realiza una ejecución de prueba y observa el funcionamiento introduciendo diferentes valores.

Vamos a plantear un caso hipotético para ver como funcionaría este código.

Imagina que el usuario tiene 25 años. El ordenador comenzaría en la línea 1, donde le preguntamos si el usuario tiene menos de 10 años. Como la respuesta es falsa pasamos a la línea 4. Aquí preguntamos si el usuario tiene menos de 20 años. En este caso también es falso, con lo que pasamos a la línea 7. Hacemos otra pregunta mirando si el usuario tiene menos de 30 años. En este caso la respuesta es Verdadera con lo que pasamos a la línea 8 donde escribimos el mensaje correspondiente en el objeto Comentario después de esto pasamos a las líneas 11, 12 y 13 donde termina el código del evento.

En el ejemplo anterior hemos visto como añadir un If dentro de otro, observa como cada If tiene su Else y su End If asociados. Observa las líneas 1, 3 y 13, las líneas 4, 6 y 12, y las líneas 7, 9 y 11. Habrás podido comprobar que hemos utilizado tabuladores en el momento de escribir el código, esto se hace para poder facilitar la lectura de las líneas de código, ya que de esta manera podemos ver con un golpe de vista, donde empieza y donde acaba un If.

Vamos a aprovechar este ejemplo para explicar otra estructura de decisión.

## PRACTICA 7 – Configura computadora de compra

Realiza un programa que permita al usuario configurar los accesorios que quiera para comprar una computadora, una vez marcados estos (con un valor constante dado en el programa), nos ira mostrando el total del costo (o valor) de la misma.

Los valores de cada accesorio los puedes poner a tu gusto.

## Ejercicio 21 – Boleta de pago (Condiciones If)

Se desea elaborar la boleta de pago de los trabajadores de una fábrica. Para ello se cuenta con los siguientes datos de entrada:

Apellidos y nombres del trabajador

Sueldo básico

Horas extras trabajadas

Además, se sabe que los trabajadores laboran en dos turnos: diurno y nocturno. Y que la tarifa por las horas extras diurnas es de 10 soles y por las horas extras nocturnas es de 15 soles.

Los descuentos a aplicar son sobre el sueldo básico y se obtienen de la siguiente manera:

$$\text{Renta} = \text{SueldoBasico} * 0.1$$

$$\text{Fonavi} = \text{SueldoBasico} * 0.07$$

$$\text{AFP} = \text{SueldoBasico} * 0.03$$

Finalmente el sueldo neto del trabajador se calcula mediante las siguientes expresiones:

$$\text{Ingresos} = \text{SueldoBasico} + \text{HorasExtras} * \text{PagoHoraExtra}$$

$$\text{Egresos} = \text{Renta} + \text{Fonavi} + \text{AFP}$$

$$\text{SueldoNeto} = \text{Ingresos} - \text{Egresos}$$

El diseño de la interfaz deberá ser similar a la figura mostrada:

**Boleta de pago**

Trabajador:

Sueldo básico S/.

Horas extras:

Pago hora extra S/.

Turno:

☐ Diurno

☒ Nocturno

Descuentos:

☒ Renta

☐ Fonavi

☒ AFP

Sueldo neto S/.



Para el desarrollo de esta aplicación, proceda a ubicar los siguientes controles en el formulario:

2 marcos

5 etiquetas

8 cajas de texto

2 botones de opción

3 casillas de verificación

3 botones de comando

Luego, proceda a establecer las propiedades según se indica a continuación:

#### **Form1**

Nombre	FrmBoletaDePago
BorderStyle	3-Fixed Dialog
Caption	Boleta de pago
Moveable	False

#### **Label1**

Nombre	LblTrabajador
AutoSize	True
Caption	Trabajador:

#### **Label2**

Nombre	LblSueldoBasico
AutoSize	True
Caption	Sueldo Bruto S/.

#### **Label3**

Nombre	LblHorasExtras
AutoSize	True

Caption	Horas extras:
---------	---------------

#### Label4

Nombre	LblPagoHoraExtra
AutoSize	True
Caption	Pago hora extra S/.

#### Label5

Nombre	LblSueldoNeto
AutoSize	True
Caption	Sueldo neto S/.

#### Text1

Nombre	TxtTrabajador
Text	

#### Text2

Nombre	TxtSueldoBasico
Text	

#### Text3

Nombre	TxtHorasExtras
Text	

#### Text4

Nombre	TxtPagoHoraExtra
BackColor	&H80000004&
Text	

**Text5**

Nombre	TxtRenta
BackColor	&H80000004&
Locked	True
Text	

**Text6**

Nombre	TxtFonavi
BackColor	&H80000004&
Locked	True
Text	

**Text7**

Nombre	TxtAFP
BackColor	&H80000004&
Locked	True
Text	

**Text8**

Nombre	TxtSueldoNeto
BackColor	&H80000004&
Locked	True
Text	

**Option1**

Nombre	OptDiurno
Caption	Diurno
Value	True

**Option2**

Nombre	OptNocturno
Caption	Nocturno
Value	False

**Check1**

Nombre	ChkRenta
Caption	Renta

**Check2**

Nombre	ChkFonavi
Caption	Fonavi

**Check3**

Nombre	ChkAFP
Caption	AFP

**Command1**

Nombre	CmdAceptar
Caption	&Aceptar
Default	True

**Command2**

Nombre	CmdLimpiar
Caption	&Limpiar

**Command3**

Nombre	CmdSalir
Cancel	True
Caption	&Salir

Una vez establecidas las propiedades, proceda a ingresar el código que se indica a continuación:

```
Private Sub CmdAceptar_Click()  
    Dim SueldoBasico As Double  
    Dim HorasExtras As Integer, PagoHoraExtra As Double  
    Dim Renta As Double, Fonavi As Double, AFP As Double  
    Dim Ingresos As Double, Egresos As Double  
    Dim SueldoNeto As Double  
    SueldoBasico = Val(TxtSueldoBasico)  
    HorasExtras = Val(TxtHorasExtras)  
    If OptDiurno Then  
        PagoHoraExtra = 10  
    End If  
    If OptNocturno Then  
        PagoHoraExtra = 15  
    End If  
    If ChkRenta.Value Then  
        Renta = SueldoBasico * 0.1  
    Else  
        Renta = 0  
    End If  
    If ChkFonavi.Value Then  
        Fonavi = SueldoBasico * 0.07  
    Else  
        Fonavi = 0  
    End If  
    If ChkAFP.Value Then  
        AFP = SueldoBasico * 0.03  
    Else  
        AFP = 0  
    End If  
    Ingresos = SueldoBasico + HorasExtras * PagoHoraExtra  
    Egresos = Renta + Fonavi + AFP  
    SueldoNeto = Ingresos - Egresos  
    TxtPagoHoraExtra = Str(PagoHoraExtra)  
    TxtRenta = Str(Renta)  
    TxtFonavi = Str(Fonavi) : TxtAFP = Str(AFP)  
    TxtSueldoNeto = Str(SueldoNeto)
```

End Sub

Private Sub CmdSalir\_Click()

End

End Sub

## Ejercicio 22 – Puntuacion de un estudiante (Condiciones If)

Cuatro notas entre 0 y 20 representan las calificaciones de un estudiante de un curso de programación. Elaborar una aplicación para obtener el promedio de esas calificaciones y visualizar su puntuación de acuerdo al siguiente cuadro:

Promedio	Puntuación
19-20	A
16-18	B
11-15	C
6-10	D
0-5	E

El diseño de la interfaz deberá ser similar a la figura mostrada:

The screenshot shows a Windows application window titled "Puntuación de un estudiante". Inside the window, there are several input fields and buttons. The "Alumno" field contains "Cáceres Larrea, Sandra". Below it are four "Nota" fields: "Nota 1" with "14.5", "Nota 2" with "17.0", "Nota 3" with "16.0", and "Nota 4" with "15.0". To the right of these fields are three buttons: "Aceptar", "Limpiar", and "Salir". At the bottom of the window, there are two more fields: "Promedio:" with the value "16" and "Puntuación:" with the value "B".

Para el desarrollo de esta aplicación, proceda a ubicar los siguientes controles en el formulario:

7 etiquetas

7 cajas de texto

3 botones de comando

Luego, proceda a establecer las propiedades según se indica a continuación:

#### **Form1**

Nombre	FrmNotas
BorderStyle	3-Fixed Dialog
Caption	Puntuación de un estudiante

#### **Label1**

Nombre	LblAlumno
AutoSize	True
Caption	Alumno:

#### **Label2**

Nombre	LblN1
AutoSize	True
Caption	Nota 1:

#### **Label3**

Nombre	LblN2
AutoSize	True
Caption	Nota 2:

#### **Label4**

Nombre	LblN3
AutoSize	True

Caption	Nota 3:
---------	---------

#### Label5

Nombre	LblN4
AutoSize	True
Caption	Nota 4:

#### Label6

Nombre	LblPromedio
AutoSize	True
Caption	Promedio:

#### Label7

Nombre	LblPuntuacion
AutoSize	True
Caption	Puntuación:

#### Text1

Nombre	TxtAlumno
Text	

#### Text2

Nombre	TxtN1
Text	

#### Text3

Nombre	TxtN2
Text	



**Text4**

Nombre	TxtN3
Text	

**Text5**

Nombre	TxtN4
Text	

**Text6**

Nombre	TxtPromedio
BackColor	&H80000004&
Locked	True
Text	

**Text7**

Nombre	TxtPuntuacion
BackColor	&H80000004&
Locked	True
Text	

**Command1**

Nombre	CmdAceptar
Caption	&Aceptar
Default	True

**Command2**

Nombre	CmdLimpiar
Caption	&Limpiar

### Command3

Nombre	CmdSalir
Cancel	True
Caption	&Salir

Una vez establecidas las propiedades, proceda a ingresar el código que se indica:

```
Private Sub CmdAceptar_Click()  
    Dim N1 As Double, N2 As Double, N3 As Double, N4 As Double  
    Dim Promedio As Integer  
    N1 = Val(TxtN1) : N2 = Val(TxtN2)  
    N3 = Val(TxtN3) : N4 = Val(TxtN4)  
    Promedio = CInt((N1 + N2 + N3 + N4) / 4)  
    TxtPromedio = Str(Promedio)  
    If Promedio >= 19 And Promedio <= 20 Then  
        TxtPuntuacion = "A"  
    ElseIf Promedio >= 16 And Promedio <= 18 Then  
        TxtPuntuacion = "B"  
    ElseIf Promedio >= 11 And Promedio <= 15 Then  
        TxtPuntuacion = "C"  
    ElseIf Promedio >= 6 And Promedio <= 10 Then  
        TxtPuntuacion = "D"  
    ElseIf Promedio >= 0 And Promedio <= 5 Then  
        TxtPuntuacion = "B"  
    Else: MsgBox "Error de datos", vbCritical, "Mensaje"  
    End If  
End Sub
```

## Ejercicio 23 – Fechas en modo completo

Escribir un aplicación que acepte fechas como tres números (dd, mm, aaaa) y las visualice del modo usual. A manera de ejemplo considere lo siguiente:



En caso de que el usuario ingrese algún dato incorrecto (por ejemplo 13 como un número de mes), se debe visualizar el siguiente mensaje:



Cuando el usuario haga click en el botón Salir, se debe observar el siguiente mensaje:



En caso de que el usuario elija la opción Si, la aplicación debe terminar. En caso contrario, es decir si el usuario elige la opción No, se debe proseguir con la aplicación.

Para el desarrollo de esta aplicación, proceda a ubicar los siguientes controles en el formulario:

4 etiquetas

3 cajas de texto

1 marco

3 botones de comando

En seguida proceda a establecer las propiedades según se indica:

#### **Form1**

Nombre	FrmFecha
BorderStyle	3-Fixed Dialog
Caption	Fecha en letras

#### **Label1**

Nombre	LblDia
AutoSize	True
Caption	Día:

#### **Label2**

Nombre	LblMes
AutoSize	True
Caption	Mes:

#### **Label3**

Nombre	LblAnno
AutoSize	True
Caption	Año:

**Label4**

Nombre	LblFecha
AutoSize	True
Caption	

**Text1**

Nombre	TxtDia
Text	

**Text2**

Nombre	TxtMes
Text	

**Text3**

Nombre	TxtAnno
Text	

**Command1**

Nombre	CmdAceptar
Caption	&Aceptar
Default	True

**Command2**

Nombre	CmdLimpiar
Caption	&Limpiar

**Command3**

Nombre	CmdSalir
Caption	&Salir

Una vez establecidas las propiedades de la interfaz proceda a ingresar el siguiente código:

```
Private Sub CmdAceptar_Click()  
    Dim CadMes As String, Mes As Integer  
    Mes = Val(TxtMes)  
    Select Case Mes  
        Case 1: CadMes = "Enero"  
        Case 2: CadMes = "Febrero"  
        Case 3: CadMes = "Marzo"  
        Case 4: CadMes = "Abril"  
        Case 5: CadMes = "Mayo"  
        Case 6: CadMes = "Junio"  
        Case 7: CadMes = "Julio"  
        Case 8: CadMes = "Agosto"  
        Case 9: CadMes = "Setiembre"  
        Case 10: CadMes = "Octubre"  
        Case 11: CadMes = "Noviembre"  
        Case 12: CadMes = "Diciembre"  
        Case Else  
            MsgBox "Error de datos...", vbCritical, "Mensaje"  
            Call CmdLimpiar_Click  
            Exit Sub  
        End Select  
        LblFecha = TxtDia & " de " & CadMes & " de " & TxtAnno  
    End Sub  
  
Private Sub CmdLimpiar_Click()  
    TxtDia = "" : TxtMes = "" : TxtAnno = ""  
    TxtDia.SetFocus  
End Sub  
  
Private Sub CmdSalir_Click()  
    If MsgBox("¿Desea terminar la aplicación?", _  
        vbQuestion + vbYesNo, "Pregunta") = vbYes Then  
        End  
    Else: Call CmdLimpiar_Click  
    End If  
End Sub
```

## PRACTICA 8 – Descuento en Restaurante

Un restaurante ofrece un descuento del 10% para consumos entre S/. 30.00 y S/. 50.00; un descuento del 20% para consumos mayores a S/. 50.00 nuevos soles; para todos los demás casos no se aplica ningún tipo de descuento. Elaborar una aplicación que permita determinar el importe a pagar por el consumidor. El diseño de la interfaz y otras consideraciones se dejan a su criterio.

### Ejercicio 24 – Select Case

Continuando con el ejercicio anterior, vamos a hacer unas modificaciones.

1. Borra el código del botón Calcular
2. Escribe en su interior el siguiente código:

---

```
Select Case Val(Edad.Text)  
  Case < 10  
    Comentario.Caption = "Menos de 10 años"  
  Case < 20  
    Comentario.Caption = "Entre 10 y 20 años"  
  Case < 30  
    Comentario.Caption = "Entre 20 y 30 años"  
  Case Else  
    Comentario.Caption = "Más de 30 años"  
End Select
```

---

3. Haz una ejercicion de prueba.

Intenta comparar los dos ultimos codigos que te hemos planteado y observa que el segundo es mucho mas claro y facil de entender que el primero en el que utilizabamos IF anidados.

## Ejercicio 25 – Select Case con operadores de comparacion

Vamos a realizar el mismo ejemplo que antes pero utilizando como operador de comparacion Mayor (>).

---

```
Select Case Val(Edad.Text)
    Case > 30
        Comentario.Caption = "Más de 30 años"
    Case > 20
        Comentario.Caption = "Entre 20 y 30 años"
    Case > 10
        Comentario.Caption = "Entre 10 y 20 años"
    Case Else
        Comentario.Caption = "Menos de 10 años"
End Select
```

---

## Ejercicio 26 – If..Then y Select Case

Vamos a realizar una practica en la que utilizaremos las dos estructuras de decision que hemos visto a lo largo de esta leccion.

Crearemos una aplicación en la que despues de seleccionar un elemento de entre una lista de objetos nos devuelva el precio. El precio podra aparecer **con** o **sin** IVA.

Para crear esta aplicación nosotros solo indicaremos que objetos se necesitan y algunas de las propiedades que se debern cambiar. El aspecto de la aplicación no es lo mas importante, pero puedes dedicarle un rato a perfeccionar la apariencia de los objetos insertados para hacer mas atractiva la aplicación.

1. Inserta un **ListBox** al que llamas ListaObjetos
2. Introduce en el LISTBOX 5 nombres de diferentes objetos que puedas comprar en cualquier tienda a los que mas tarde pondremos precio.
3. Inserta un **Frame**.
4. Dentro de este Frame inserta dos **OptionButton**.
5. El primero tendra como nombre **ConIVA** y como Caption: Con IVA.
6. El segundo tendra como nombre **SinIVA** y como Caption: Sin IVA.
7. Activa uno de los dos OptionButton que has insertado anteriormente

Según el OPTIONBUTTON que se seleccione se mostrara el precio con o sin IVA. Imaginemos que el IVA (Impuesto al Valor Agregado) es de un 16% sobre el precio del producto.



8. Inserta un boton al que llamaremos **MostrarPrecio** y como Caption tendra Mostrar Precio. Al pulsar dicho boton la aplicación nos enseñara el precio del producto señalado.
9. Inserta un **Label** al que le borraremos el contenido y le pondremos como nombre **Precio**. En dicho Label mostraremos el precio del producto seleccionado.

Empezando a codificar.

Para saber cual de los objetos de la lista esta activado vamos a utilizar la propiedad de **ListIndex** del objeto LISTAOBJETOS, la cual nos devuelve el indice del elemento seleccionado. Recuerda que dicho indice siempre empieza a contar desde 0.

Vamos a utiliza la estructura de decision Select Case, la cual nos permitira tomar un camino u otro en el codigo según el indice que este seleccionado de nuestra lista.

Observa la estructura del codigo que mas adelante escribiremos en el evento Click del boton MostrarPrecio.

---

```
Select Case ListaObjetos.ListIndex  
  Case 0  
    ` Instrucciones primer objeto seleccionado.  
  Case 1  
    ` Instrucciones segundo objeto seleccionado.  
  Case 2  
    ` Instrucciones tercer objeto seleccionado.  
  Case 3  
    ` Instrucciones cuarto objeto seleccionado.  
  Case 4  
    ` Instrucciones quinto objeto seleccionado.  
End Select
```

---

Las líneas que tienen este símbolo al principio . solo son líneas de comentario. El ordenador en el momento de ejecutar este código las pasaría por alto. Es recomendable utilizar los comentarios para así facilitar la lectura del código.

El ordenado al pulsar el botón, miraría cual es el índice de la tabla que está seleccionado y ejecutaría las líneas de instrucciones que están dentro del índice indicado. En dichas líneas podemos hacer que el ordenador mire si tenemos seleccionado el precio con o sin Iva y nos lo muestre. También podríamos hacer que el precio de dicho objeto seleccionado lo guarde en una variable y después de la estructura Select Case mire si está seleccionado el Iva o no y se realicen los cálculos pertinentes en cada uno de los casos.

Vamos a poner los dos códigos y después miraremos cual de ellos es más correcto o más útil.

Primero escribiremos el código utilizando un If dentro de cada uno de los objetos seleccionados. Puedes poner los precios que tú quieras dependiendo de los objetos que hayas escrito dentro de la lista.

10. Completa el código de la siguiente manera.

---

```
Private Sub MostrarPrecio_Click()  
  Select Case ListaObjetos.ListIndex  
    Case 0  
      If SinIVA.Value = True Then  
        Precio.Caption = 1000  
      Else  
        Precio.Caption = (1000 * 16 / 100) + 1000  
      End If  
    Case 1  
      If SinIVA.Value = True Then  
        Precio.Caption = 2000  
      Else  
        Precio.Caption = (2000 * 16 / 100) + 2000  
      End If  
    Case 2  
      If SinIVA.Value = True Then  
        Precio.Caption = 3000  
      Else  
        Precio.Caption = (3000 * 16 / 100) + 3000  
      End If  
    Case 3  
      If SinIVA.Value = True Then  
        Precio.Caption = 4000  
      Else  
        Precio.Caption = (4000 * 16 / 100) + 4000  
      End If  
    Case 4  
      If SinIVA.Value = True Then  
        Precio.Caption = 5000  
      Else  
        Precio.Caption = (5000 * 16 / 100) + 5000  
      End If  
  End Select  
End Sub
```

---

Observa que el IVA lo calcula la computadora, multiplicando el precio por un 16% (16/100) y sumandoselo al precio de dicho objeto.

11. Realiza una ejecucion de prueba seleccionando diferentes objetos, indicando si deseas ver el precio con o sin IVA.

Ahora vamos a poner el mismo código utilizando la estructura IF una sola vez y utilizando una variable.

---

```
Private Sub MostrarPrecio_Click()  
  Select Case ListaObjetos.ListIndex  
    Case 0  
      Precios = 1000  
    Case 1  
      Precios = 2000  
    Case 2  
      Precios = 3000  
    Case 3  
      Precios = 4000  
    Case 4  
      Precios = 5000  
  End Select  
  If SinIVA.Value = True Then  
    Precio.Caption = Precios  
  Else  
    Precio.Caption = Precios + (Precios * 16 / 100)  
  End If  
End Sub
```

---

Si te fijas para el primero de los dos códigos hemos utilizado muchas más líneas que para el segundo. Esto es debido a que en cada uno de los casos producidos por el índice de la lista poníamos una estructura If, para comprobar si teníamos seleccionado el precio con o sin IVA. En el segundo ejemplo solo ponemos la estructura If al final del código justo antes de mostrar el precio. En cada uno de los casos almacenamos el precio del objeto seleccionado en una variable para después poder calcular con él el precio total del objeto.

Tanto como en entendimiento, como en cantidad de líneas que ocupa el código es mucho mejor utilizar el segundo caso que no el primero. Siempre tenemos que evitar escribir líneas que puedan estar repetidas o que podamos evitarlas de alguna forma.

## Ejercicio 27 – For...Next

Vamos a realizar una simple aplicación en el que utilizaremos una estructura de repetición utilizando un contador.

La aplicación consistirá en una simulación de una tirada de un dado.

Te iremos especificando que tipo de objetos deberás añadir en nuestro formulario y algunas de las propiedades que deberás cambiar. El aspecto de los objetos y su situación corren por tu cuenta. Puedes poner tantos objetos Label como quieras para aclarar para que sirven cada uno de los elementos insertados en el formulario.

1. Sitúa en un formulario nuevo un ListBox al que deberás poner como (Nombre): Dado.

Aquí será donde el ordenador nos muestre las diferentes tiradas que realizamos.

2. Coloca un CommandBotton, que tendrá como (Nombre) y Caption: Tirada. Al pulsar este botón se realizarán las diferentes tiradas.

3. Coloca un TextBox al que pondremos como (Nombre): NumTiradas. Borra el contenido que aparece por defecto dentro de este objeto.

Aquí será donde indiquemos cuantas tiradas queremos realizar.

Una vez colocados los objetos vamos a pensar en el código.

Nosotros en esta práctica queremos que se realicen tantas tiradas de dado como nos indique el usuario dentro del TextBox. Para esto nos interesaría crear una estructura de repetición que debería empezar en 1 y terminar en el número que indica el usuario. Los incrementos que sufrirá el contador deberá ser de uno en uno, por lo que la parte del step no la especificaremos.

4. Haz doble clic dentro del botón y escribe el siguiente código.

---

```
For Contador = 1 To NumTiradas.Text  
Dado.AddItem (Int(6*Rnd)+1)  
Next Contador
```

---

En la primera línea de este pequeño código, que más adelante depuraremos, hemos iniciado el contador (nueva variable) a 1. No hace falta que definamos la variable. Al no definirla esta es de tipo Variant<sup>1</sup>.

En esta primera línea también definimos en que valor queremos que termine el bucle. Este valor será el valor que introduzca el usuario dentro del TextBox.

En la segunda línea, Dado.AddItem (Int(6\*Rnd)+1)), hacemos que Visual Basic nos busque un valor aleatorio. Esto lo conseguimos con la instrucción Rnd. Nosotros como queremos conseguir un número aleatorio dado un intervalo, del 1 al 6 (valores que tiene un dado común), necesitamos utilizar una estructura determinada:

$\text{Int}([\text{Valor superior}] \cdot [\text{Valor inferior}] + 1) * \text{Rnd} + [\text{Valor inferior}]$

Valor inferior: nos indica el valor mínimo que tiene el intervalo.

Valor superior: nos indica el valor máximo del intervalo.

En nuestro ejemplo esto quedaría de la siguiente manera. Recuerda que queremos valores enteros, por eso utilizamos la instrucción `Int(Valor)`, entre el 1 y el 6.

`Int(6-1+1*Rnd)+1` Resolviendo las operaciones la instrucción quedaría de la siguiente forma

`Int(6*Rnd)+1`. Con esto conseguiríamos números aleatorios entre el 1 y el 6, ambos inclusive.

Pongamos otro ejemplo: imaginemos que ahora queremos obtener valores aleatorios entre el 10 y el 20. La instrucción quedaría de la siguiente forma. `Int(20-10+1*Rnd)+10` resolviendo las operaciones, la instrucción quedaría así:

`Int(11*rnd)+10`.

Para añadir elementos a una lista deberemos utilizar la instrucción `AddItem`. Cada vez que se pasa por esta línea se inserta un nuevo elemento ocupando el puesto de último índice + 1. Recuerda que el primer elemento ocuparía la posición con índice 0. La estructura de esta instrucción es la siguiente:

`[Nombre de la lista].AddItem [Cadena a añadir]`

Nombre de la lista: es el nombre que le hemos puesto a la lista donde queremos que se añadan los diferentes elementos.

Cadena a añadir: es el valor, cadena, variable... que queremos añadir a nuestra lista.

Observa que esta instrucción, aunque se trate de una asignación, no utiliza el signo igual.

En nuestra aplicación queremos añadir el valor aleatorio obtenido anteriormente. Así que la línea de código quedará de la siguiente manera: `Dado.AddItem (Int(6*Rnd)+1)`

Observaciones con números aleatorios

5. Inicia una ejecución de prueba.
6. Indica que quieres realizar 5 tiradas.
7. Pulsa sobre el botón: Tirada.
8. Observa con detenimiento la secuencia de números que han aparecido en la lista.
9. Detén la ejecución del programa.
10. Vuelve a ejecutar el programa.
11. Indica que quieres realizar nuevamente 5 tiradas.
12. Pulsa sobre el botón: Tirada.
13. Observa la secuencia de tiradas de la lista.

Si recuerdas la primera secuencia que ha aparecido en nuestra primera ejecución y la comparas con la actual, podrás ver que es exactamente igual. Esto es debido a que, mientras no indiquemos lo contrario, la secuencia de números aleatorios obtenidos con Rnd siempre será la misma. Como podrás ver esto no nos interesa en la gran mayoría de casos, con lo que utilizaremos una nueva instrucción que nos permitirá obtener valores completamente aleatorios.

14. Detén la ejecución del programa.

Vamos a ver una manera para que cada vez que se inicia el programa los valores que se consiguen con la instrucción Rnd sean diferentes.

15. Pulsa doble clic sobre el fondo del formulario.

Te aparecerá la ventana de código con un nuevo evento Form\_Load(). Este evento se ejecuta justo en el momento en el que se carga el formulario. En este caso, como solo disponemos de un formulario, este evento se ejecutará al poner en funcionamiento la aplicación.

16. Escribe dentro de dicho evento Randomize.

Esta instrucción nos sirve para iniciar con valores, cada vez diferentes, la secuencia de números aleatorios. De tal forma que cada vez que ejecutemos nuestra aplicación obtendremos secuencias aleatorias diferentes.

17. Vuelva a realizar los pasos del 5 al 13. Pero esta vez observa como la primera y la segunda secuencia son diferentes.

18. Sin detener la ejecución del programa, vuelve a pedir que se realicen 5 tiradas más.

Observa como en la lista se han añadido 5 valores más a los que ya teníamos, de esta forma ahora tenemos 10 valores (5 tiradas anteriores y 5 actuales). Si nosotros seguimos realizando tiradas, los valores de las nuevas tiradas se van añadiendo a la lista de forma indefinida. Observa que cuando la cantidad de valores superan el tamaño de la lista aparece una barra de desplazamiento vertical que nos permite poder visualizar los valores que hemos conseguido en tiradas anteriores.

Si deseas ver los valores de la lista, en lugar de en filas en columnas deberías acceder a la propiedad Columns de la lista y cambiar el número de columnas que deseas ver. Si modificas este valor, en el momento que tengamos más elementos de los que caben en la lista aparecerá una barra de desplazamiento horizontal en lugar de vertical. Prueba esta propiedad.

A nosotros, en esta práctica, lo que nos interesaría es conseguir que cada vez que se realice una nueva tirada se borre el contenido de la tabla y aparezcan las nuevas tiradas.

Vamos a ver como podemos borrar la lista cada vez que realizamos una tirada nueva.

19. Detén la ejecución del programa.
20. Pulsa doble clic en el botón: Tirada.
21. Completa el código que ya tienes, para que quede de la siguiente forma:

---

```
Dado.Clear  
For Contador = 1 To NumTiradas.Text  
    Dado.AddItem (Int(6*Rnd)+1)  
Next Contador
```

---

La instrucción Clear sirve para borrar el contenido de la tabla. La sintaxis de esta instrucción es la siguiente: [Nombre lista].Clear.

De esta forma cada vez que queramos realizar una nueva tirada, primero se borrará el contenido de la lista y después se añadirán los elementos nuevos. Cada vez que se borran los elementos de la lista, el índice de la lista vuelve a tener como valor 0.

## Ejercicio 28 – For...Next con validacion numerica

En este apartado vamos a hacer que el usuario solo pueda poner números en el número de tiradas que quiere realizar, y no pueda introducir ningún tipo de carácter más. Esto es una medida de depuración del programa, ya que de esta forma evitamos que la aplicación aborte al producirse un error.

Veamos que ocurre si introducimos una letra en el número de tiradas deseadas.

1. Ejecuta la aplicación.
2. Escribe una letra en la casilla para indicar el número de tiradas que desees realizar.
3. Pulsa el botón: Tirada.

Observa como te aparece una ventana indicando que se ha producido un error. El mensaje de error es: No coinciden los tipos. Esto quiere decir que Visual Basic no ha podido utilizar lo que nosotros hemos escrito en el interior del número de tiradas como contador para nuestro bucle. Visual Basic necesita un número y no una letra.

4. Pulsa el botón Terminar, que aparece en la pantalla de error.

De esta forma podemos volver a la edición del código. Esta ventana de error es la que tendremos que evitar en muchos casos, para que el usuario no se encuentre con la aplicación colgada.

5. Haz doble clic sobre el TextBox.

Observa como el evento que se ha abierto ha sido Change. El código que escribimos dentro de este evento se ejecutará en el momento en el que se produce un cambio dentro del TextBox.

6. Abre la lista desplegable de los eventos de este objeto y selecciona KeyPress.

Fíjate como ha aparecido un nuevo procedimiento: Private Sub NumTiradas\_KeyPress(KeyAscii As Integer). La parte que se encuentra dentro de los paréntesis, devuelve al procedimiento un valor KeyAscii siendo este un valor numérico que representa la tecla que se ha pulsado. Esta tabla tiene 256 elementos numerados del 0 al 255, y cada uno de ellos representa un carácter diferente.

7. Inserta estas líneas de código dentro de KeyPress.

---

```
If (KeyAscii < 48 Or KeyAscii > 57) Then  
  If (KeyAscii <> 8) Then KeyAscii = 0  
End If
```

---

Con estas líneas de código, conseguiremos que el usuario en el momento de pulsar alguna tecla que no sea un número no se escriba dentro del TextBox.

8. Realiza una ejecución de prueba e intentar escribir alguna letra. Observa como no se escribe nada en el interior de este objeto.

9. Escribe cualquier valor numérico y pulsa Tirada.

10. Detén la ejecución del programa.

## Ejercicio 29 – Do...Loop

Vamos a realizar una práctica en la que intentaremos ordenar una tabla de elementos que inicialmente están desordenados. Para ordenar una tabla existen multitud de métodos diferentes. Algunos de ellos muy simples y poco eficaces, otros son complejos y con un alto grado de eficacia. La dificultad del sistema de ordenación la escogeremos según la cantidad de elementos que deseamos ordenar.

En nuestro caso realizaremos una aplicación que nos ordenará una pequeña tabla que contiene datos aleatorios. Utilizaremos el método de ordenación más sencillo y menos eficaz. Este método, llamado Método de la burbuja, es ideal para tablas con pocos datos.

Método de la burbuja

El método de la burbuja se basa en el intercambio de elementos de dos en dos.



Si nosotros queremos ordenar la tabla ascendentemente, el intercambio de los elementos se produce cuando el primero de ellos es mayor que el segundo. Repitiendo este proceso por lo largo de la tabla conseguimos que el elemento más grande pase a estar en el último lugar de la tabla. El elemento sube por la tabla hasta que ocupa la posición más alta. De ahí viene el nombre de ordenación de la burbuja, el elemento sube como si se tratase de una burbuja dentro de un recipiente.

Los pasos que se siguen exactamente en esta ordenación son los siguientes:

1. Se compara el primer elemento con el segundo de la tabla. Si están desordenados (el primero es más grande que el segundo, en el caso de la ordenación ascendente) se intercambian. Luego comparamos el segundo con el tercero, si es necesario los intercambiamos. Continuamos con los intercambios hasta que comparamos el penúltimo con el último.
2. Como segundo paso, volvemos a repetir el primero pero esta vez hasta llegar a comparar el antepenúltimo con el penúltimo, ya que el último elemento ya está ordenado gracias al primer paso.
3. Volvemos a repetir exactamente lo mismo que en el paso uno, pero esta vez con un elemento menos, ya que los dos últimos ya están ordenados.

Este método termina en el momento en el que hemos hecho tantas pasadas como elementos menos 1 hay en la lista. Realizamos una pasada menos de la cantidad de elementos que hay en la tabla, ya que si todos los elementos de la tabla se han ido ordenando según hemos pasado, como es lógico este último elemento a ordenar ya estará ordenado.

Vamos a realizar esta aplicación. Como en todas las prácticas sigue los pasos que te indicamos.

Insertar los elementos

1. Abre un proyecto nuevo.
2. Inserta un `CommandButton` al que le pondrás como (Nombre): Nueva. Cambia su `Caption` y escribe Nueva. Este botón servirá para borrar la tabla que tengamos en pantalla y crear otra. Para poder visualizar las tablas que vallamos creando utilizaremos un `ListBox`.
3. Inserta un `ListBox`. Cambia su tamaño hasta llegar aproximadamente a 855 x 3180..
4. Cambia el (Nombre) de este `ListBox` por Lista. No introduzcas nada en su interior.
5. Inserta otro `CommandButton`. A este llámale Ordenar y ponle como `Caption`: Ordenar.

Al pulsar este botón realizaremos la ordenación de la tabla y la visualizaremos en nuestro `ListBox`. Recuerda que puedes cambiar todas las propiedades que desees de los objetos insertados en este formulario.

Vamos a definir la tabla en la que guardaremos todos los valores.

Vamos a pensar como crear esta aplicación para que sea fácil de modificar en el momento en el que deseemos cambiar el número de elementos que componen la tabla.

Para ello vamos a crear una constante que utilizaremos a lo largo del programa. En el momento que deseemos utilizar una tabla con más o menos elementos y que el programa funcione exactamente igual, solo deberemos cambiar el valor de esta constante.

6. Accede al apartado General - Declaraciones de nuestra página de código y escribe lo siguiente:

---

**Const Elementos = 12**

---

Con esto crearemos una constante llamada Elementos que podremos consultar a lo largo de todo nuestro programa.

Vamos a crear en el mismo apartado una tabla que tenga el número de elementos que marca la constante anteriormente creada. Además esta tabla, para facilitar la comprensión de nuestro código, pondremos como primer elemento el número 1 y como último Elemento.

7. Escribe la siguiente línea de código a continuación de la que ya teníamos:

---

**Dim Tabla(1 To Elementos) As Integer**

---

Observa la declaración del tamaño de la tabla, desde el elemento número 1 al elemento Elementos. La tabla la hemos definido como Integer (números enteros).

Iniciar proyecto

Vamos a escribir el código necesario para que al iniciar el proyecto nos aparezca una tabla de número aleatorios dentro de nuestra tabla.

8. Pulsa un clic en el fondo del escritorio.

Seguidamente te aparecerá la ventana de código. Observa el evento que tenemos abierto. Private Sub Form\_Load() Esto nos indica que todo lo que escribamos dentro de este evento se realizará en el momento en el que carguemos el formulario.

9. Copia las siguientes instrucciones dentro de dicho evento.

---

```
Private Sub Form_Load()  
  Randomize  
  For Contador = 1 To Elementos  
    Tabla(Contador) = Int((9 * Rnd) + 1)  
    Lista.AddItem Tabla(Contador)  
  Next Contador  
End Sub
```

---

Observa que en la primera línea, dentro del evento, hemos escrito la instrucción `Randomize` para iniciar la función de números aleatorios.

En la segunda línea hemos iniciado un bucle que se repetirá hasta que `Contador` llegue hasta el número de elementos que hemos definido en `Elementos`. Observa como la variable `Contador` no la hemos definido anteriormente.

Dentro de este bucle se realizará el relleno de los elementos de nuestra `Tabla` con números aleatorios generados mediante la instrucción `Int((9 * Rnd) + 1)`. Observa que para movernos por la tabla utilizamos como índice, nuestro `Contador`. A la vez que llenamos la tabla vamos añadiendo a nuestra `Lista` los elementos que se acaban de crear. De esta manera a la vez que los creamos los pasamos a la lista, así no tenemos que volver a realizar otra pasada por la tabla.

10. Cierra la ventana de código.

11. Haz doble clic en el botón `Nueva`.

Como ya hemos dicho, en el momento en el que pulsemos este botón borraremos lo que haya en la `Lista` e introduciremos una nueva tabla.

12. Sube por el código de la aplicación hasta llegar al evento que hemos escrito anteriormente.

Copiar y pegar

13. Selecciona el contenido de dicho evento.

Para seleccionar líneas de código, simplemente debes ponerte en el margen izquierdo de la ventana de código a la altura de la primera línea de código. Seguidamente pulsa el botón izquierdo del ratón y mientras lo tienes pulsado muévete hasta la última la ultima línea de código dentro de este procedimiento. Fíjate como ha cambiado el color del fondo del texto.

Con el texto seleccionado:

14. Selecciona la opción `Copiar` del menú `Edición`.

15. Sitúa el cursor en el interior del evento del botón `Nueva`.

16. Selecciona la opción `Pegar` del menú `Edición`.

El código se ha copiado.

Vamos a realizar unas pequeñas modificaciones.

17. Selecciona la primera línea de código y bórrala.

18. Escribe lo siguiente: `Lista.Clear`

Recuerda que esta instrucción sirve para borrar el contenido de la Lista para así poder poner insertar otra lista nueva.

19. Realiza una ejecución de prueba para ver el funcionamiento de los dos eventos programados hasta el momento.

20. Detén la ejecución.

### Ordenación

Ahora vamos a dedicarnos a lo que es en sí la ordenación de la Tabla.

21. Haz doble clic sobre el botón Ordenar.

22. Escribe dentro de este evento las siguientes instrucciones.

---

```
1 Private Sub Ordenar_Click()  
2     I = 1  
3     Do  
4         For J = 1 To Elementos - I  
5             If Tabla(J) >= Tabla(J + 1) Then  
6                 Cambio = Tabla(J)  
7                 Tabla(J) = Tabla(J + 1)  
8                 Tabla(J + 1) = Cambio  
9             End If  
10        Next J  
11        I = I + 1  
12    Loop Until I > (Elementos - 1)  
13    Lista.Clear  
14    For Contador = 1 To Elementos  
15        Lista.AddItem Tabla(Contador)  
16    Next Contador  
17 End Sub
```

---

Los números que aparecen en cada línea no debes copiarlos, los utilizaremos para facilitar la explicación del funcionamiento del código.

En la línea 2 iniciamos una variable llamada `I` a 1. Esta variable nos controlará, dentro de un bucle que crearemos en líneas consecutivas, las veces que tenemos que recorrer la tabla, para que esté completamente ordenada. El número de veces será: el número de elementos de la tabla menos 1.

En la línea 3 escribimos la primera línea de nuestra estructura `Do...Loop` que termina en la 12. Utilizamos una estructura `Loop Until` ya que deseamos que se repita este bucle mientras no se cumpla la condición. Recuerda que en este tipo de estructuras la condición está en la última línea del bucle.

En la línea 4 iniciamos otro bucle, en este caso un For... Next ya que nos interesa que se repitan unas serie de instrucciones un número de veces determinado.

En esta línea definimos una nueva variable llamada J con valor 1. Esta variable es la encargada de controlar el bucle. Este bucle se repetirá hasta que J llegue al valor que se le indica después del To. En nuestro caso cada vez que se ejecute este bucle se repetirá hasta un número diferente, ya que deberemos llegar hasta Elementos . I.

Vamos a explicarlo con un ejemplo: imaginemos una tabla con 5 elementos, el ordenador la primera vez que entre en el bucle principal la variable I tendrá como valor 1.

Entonces el segundo bucle deberá repetirse hasta que J llegue a 5 menos el valor de I que es 1, por lo tanto 4. Así nos aseguramos que al comparar el elemento que nos marca J con el siguiente, en la línea 5 no nos pasemos del índice de la tabla produciéndose un desbordamiento.

En la línea 5 realizamos la comparación del elemento de la tabla cuyo índice es realizamos el cambio de los valores, haciendo pasar el valor de Tabla(J+1) a Tabla(J).

Este cambio lo efectuamos de la siguiente manera en las líneas 6, 7 y 8.

En la línea 6 acumulamos el valor de Tabla(J) en una nueva variable que utilizaremos de puente entre las dos posiciones de la tabla. A esta variable puente la llamaremos Cambio.

En la línea 7 pasamos el contenido de Tabla(J+1) a Tabla(J) con lo que escribimos encima del valor que esta tenía, pero no importa ya que este valor está copiado en la variable Cambio.

En la línea 8 completamos el cambio pasando el contenido de la variable Cambio a Tabla(J+1), machacando el valor antiguo que ya está copiado en Tabla(J).

En la línea 12 termina nuestro Do... Loop Until. Cuando se ejecuta la línea 13 la tabla ya debe estar completamente ordenada, con lo que podemos pasar a visualizarla en nuestro ListBox.

Para que no se mezclen la tabla ordenada con la desordenada, primero es preferible borrar la lista, para ello utilizamos la instrucción Lista.Clear.

Para pasar el contenido de la tabla a la lista utilizamos un nuevo bucle (línea 14 y 16), utilizando la variable Contador que vuelve a tomar como primer valor 1 y como último el número de elementos de la tabla definido por la constante definida en un principio.

Al pasar por la línea 15 el contenido de la tabla en la posición que nos indica el contador pasa a añadirse a la lista. De esta forma volvemos a ver el contenido de la lista que en este caso ya está completamente ordenada.

23. Realiza una ejecución de prueba.

24. Observa los valores de la lista.
25. Pulsa en el botón Ordenar.
- Observa como en un breve espacio de tiempo vuelve a aparecer los mismos valores, pero esta vez completamente ordenados.
26. Pulsa en el botón Nueva.
27. Vuelve a ordenar la lista.
28. Detén la ejecución del programa.
29. Vuelve al modo diseño.
30. Accede a la línea en la que se le asigna un valor a la constante Elementos.
31. Modifica este valor, poniendo 5.
32. Realiza otra ejecución de prueba.

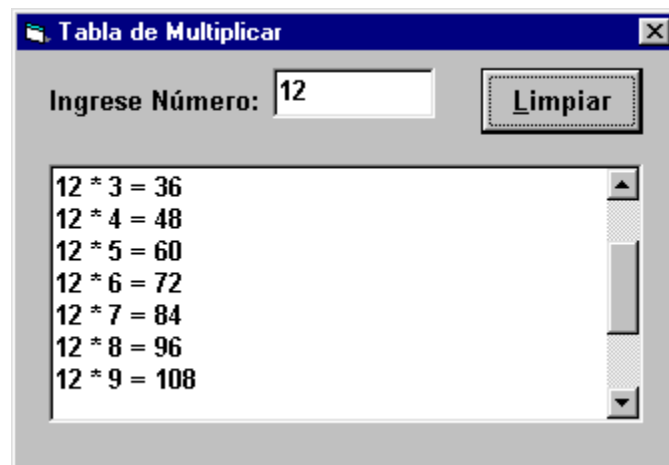
Observa como en esta ocasión solo aparecen 5 elementos en la lista. Esto a sido gracias a que en todo nuestro código utilizábamos una constante que controlaba el número de elementos que deseábamos que aparecieran en nuestra tabla. Mira cuantas líneas de código hubiésemos tenido que cambiar si no hubiésemos utilizado una constante.

33. Detén la ejecución.
34. Graba el formulario y el proyecto.

Antes de seguir adelante vamos a aprovechar el ejemplo que acabamos de realizar para explicar un elemento que podemos utilizar en muchas ocasiones el cual nos facilitará un poco la tarea de programar.

## Ejercicio 30 – Tablas de multiplicar

Escribir una aplicación que lea un número entero y muestre la tabla de multiplicar de dicho número. El diseño de entrada y salida debe ser similar al siguiente:



Para el desarrollo de esta aplicación, proceda a ubicar los siguientes controles en el formulario:

1 etiqueta

2 cajas de texto

1 botón de comando

En seguida proceda a establecer las propiedades según se indica:

#### **Form1**

Nombre	FrmTabla
BorderStyle	3-Fixed Dialog
Caption	Tabla de multiplicar

#### **Label1**

Nombre	LblNumero
Caption	Ingrese un número:

#### **Text1**

Nombre	TxtNumero
Text	

#### **Text2**

Nombre	TxtTabla
MultiLine	True
Locked	True
ScrollBars	2-Vertical
Text	

#### **Command1**

Nombre	CmdLimpiar
Caption	&Limpiar

Una vez diseñada la interfaz, proceda a ingresar el código que se indica a continuación:

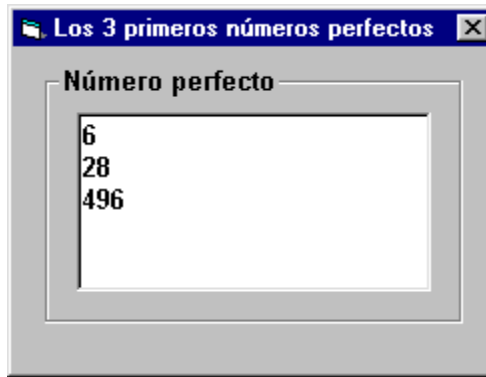
```
Private Sub TxtNumero_Change()  
    If IsNumeric(TxtNumero) Then  
        Dim N As Integer, P As Integer, I As Integer  
        Dim S As String  
        N = Val(TxtNumero)  
        S = ""  
        For I = 0 To 12  
            P = N * I  
            S = S & N & " * " & I & " = " & P & vbCrLf  
        Next I  
        TxtTabla = S  
    ElseIf TxtNumero = "" Then  
        Exit Sub  
    Else: MsgBox "Ingrese un número", vbCritical, "Mensaje"  
        TxtTabla = ""  
    End If  
End Sub  
  
Private Sub CmdLimpiar_Click()  
    TxtNumero = "" : TxtTabla = ""  
    TxtNumero.SetFocus  
End Sub
```

## Ejercicio 31 – Numeros primos

Un número perfecto es un entero positivo, que es igual a la suma de todos los enteros positivos (excluido el mismo) que son divisores del número. El primer número perfecto es 6, ya que los divisores de 6 son 1, 2, 3 y  $1 + 2 + 3 = 6$ . Escribir una aplicación que encuentre los tres primeros números perfectos.

El diseño de la interfaz debe ser similar a la figura mostrada:





Para el desarrollo de esta aplicación, proceda a ubicar los siguientes controles en el formulario:

1 marco

1 caja de texto

En seguida proceda a establecer las propiedades según se indica:

#### Form1

Nombre	FrmNumeroPerfecto
BorderStyle	3-Fixed Dialog
Caption	Los 3 primeros números perfectos

#### Frame1

Nombre	FraPerfecto
Caption	Número perfecto

#### Text1

Nombre	TxtPerfecto
MultiLine	True
Text	

Una vez establecidas las propiedades proceda a ingresar el código que se indica a continuación:

```

Private Sub Form_Load()
    Dim N As Long, I As Long, S As Long
    Dim K As Integer, Cad As String
    N = 1 : K = 0 : Cad = ""
    While True
        S = 0
        For I = 1 To (N - 1)
            If N Mod I = 0 Then S = S + I
        Next I
        If N = S Then
            Cad = Cad & N & vbCrLf
            K = K + 1
        End If
        If K = 3 Then
            TxtPerfecto = Cad
            Exit Sub
        End If
        N = N + 1
    Wend
End Sub

```

## Ejercicio 32 – Cifras y divisores de un numero

Construya una aplicación que permita el ingreso de un número entero y muestre en pantalla la siguiente información: 1) Cantidad de cifras, 2) Suma de cifras impares, 3) Suma de cifras pares, 4) Suma total de cifras, 5) Cifra mayor, 6) Cifra menor y 7) Divisores de dicho número.

El diseño de la interfaz debe ser similar a la figura siguiente:

Para el desarrollo de esta aplicación, proceda a ubicar los siguientes controles en el formulario:

4 marcos

7 etiquetas

8 cajas de texto

1 botón de comando

En seguida proceda a establecer las propiedades según se indica:

### Form1

Nombre	FrmNumeroPerfecto
BorderStyle	3-Fixed Dialog
Caption	Los 3 primeros números perfectos
Moveable	False
StartPosition	2-CenterScreen

**Frame1**

Nombre	FraEntrada
Caption	

**Frame2**

Nombre	FraSalida
Caption	

**Frame3**

Nombre	FraDivisores
Caption	Divisores

**Frame4**

Nombre	FraSalir
Caption	

**Label1**

Nombre	LblNumero
AutoSize	True
Caption	Ingrese un número:

**Label2**

Nombre	LblCantCifras
AutoSize	True
Caption	Cantidad de cifras:

**Label3**

Nombre	LblSumImpares
--------	---------------

AutoSize	True
Caption	Suma de cifras impares:

#### Label4

Nombre	LblSumPares
AutoSize	True
Caption	Suma de cifras pares:

#### Label5

Nombre	LblSumTotal
AutoSize	True
Caption	Suma total de cifras:

#### Label6

Nombre	LblCifraMayor
AutoSize	True
Caption	Cifra mayor:

#### Label7

Nombre	LblCifraMenor
AutoSize	True
Caption	Cifra menor:

#### Text1

Nombre	TxtNumero
Text	

#### Text2

Nombre	TxtCantCifras
Locked	True

Text	
------	--

### Text3

Nombre	TxtSumImpares
Locked	True
Text	

### Text4

Nombre	TxtSumPares
Locked	True
Text	

### Text5

Nombre	TxtSumTotal
Locked	True
Text	

### Text6

Nombre	TxtCifraMayor
Locked	True
Text	

### Text7

Nombre	TxtCifraMenor
Locked	True
Text	

### Text8

Nombre	TxtDivisores
MultiLine	True
Locked	True

ScrollBars	2-Vertical
Text	

### Command3

Nombre	CmdSalir
Caption	&Salir
Picture	C:\FundVB\Bitmaps\Exit.bmp
Style	1-Graphical

Una vez establecidas las propiedades proceda a ingresar el código que se indica a continuación:

```

Private Sub CmdAceptar_Click()
  If IsNumeric(TxtNumero) Then
    Dim S As Integer, SI As Integer, SP As Integer
    Dim May As Integer, Min As Integer
    Dim Cad As String
    Dim I As Integer, J As Integer
    N = CLng(TxtNumero)
    M = CLng(TxtNumero)
    Cad = ""
    I = 0
    J = 1
    S = SP = SI = 0
    For J = 1 To N
      If (N Mod J = 0) Then
        Cad = Cad & J & vbCrLf
      End If
    Next J
    While (N > 0)
      If ((N Mod 10) Mod 2) = 0 Then
        SP = SP + (N Mod 10)
      Else
        SI = SI + (N Mod 10)
      End If

```

```

    S = S + (N Mod 10)
    N = N \ 10
    I = I + 1
Wend
May = Mid(TxtNumero, 1, 1)
Men = May
While (M > 0)
    If May < (M Mod 10) Then
        May = M Mod 10
    End If
    If Men > (M Mod 10) Then
        Men = M Mod 10
    End If
    M = M \ 10
Wend
TxtCantCifras = Str(I)
TxtSumImpares = Str(SI)
TxtSumPares = Str(SP)
TxtSumTotal = Str(S)
TxtCifraMayor = Str(May)
TxtCifraMenor = Str(Men)
TxtDivisores = Cad
Else
    MsgBox "Debe ingresar un número", vbCritical, "Mensaje"
    TxtNumero.SetFocus
End If
End Sub

Private Sub CmdSalir_Click()
    If MsgBox("¿Desea terminar la aplicación?", _
        vbQuestion + vbYesNo, "Pregunta") = vbYes Then

        End
    Else
        Cancel = True
        TxtNumero.SetFocus
    End If
End Sub

```



## PRACTICA 9 – Centro numerico en una lista

Un centro numérico es un número que separa una lista de números enteros (comenzando en 1) en dos grupos de números, cuyas sumas son iguales. El primer centro numérico es el 6, el cual separa la lista (1 a 8) en los grupos: (1; 2; 3; 4; 5) y (7; 8) cuyas sumas son ambas iguales a 15. El segundo centro numérico es el 35, el cual separa la lista (1 a 49) en los grupos: (1 a 34) y (36 a 49) cuyas sumas son ambas iguales a 595. Se pide elaborar una aplicación que calcule los centros numéricos entre 1 y N.

El diseño de la interfaz y otras consideraciones se dejan a su criterio.

### Ejercicio 33 – Funciones o Procedimientos

Si observamos el ejemplo que acabamos de realizar podremos observar como hay unas cuantas líneas que se repiten en dos eventos diferentes.

---

```
For Contador = 1 To Elementos  
  Tabla(Contador) = Int((9 * Rnd) + 1)  
  Lista.AddItem Tabla(Contador)  
Next Contador
```

---

Estas líneas están dentro de los eventos: Form\_Load() y Nueva\_Click().

En esta ocasión no ocurre no es importante que estas líneas se repitan ya que son sólo 4. Pero podemos tener otras aplicaciones en las que el número de líneas que se repitan puedan ser muchas más, con lo que el número total de líneas de código se vería incrementado haciendo más difícil la localización de un posible error.

Vamos a ver una forma de poder compartir estas líneas de código y utilizarlas en el momento en el que deseemos.

35. Selecciona la opción: Agregar procedimiento dentro de la opción Herramientas.

Seguidamente te aparecerá una ventana como la siguiente:

De esta nueva ventana vamos a explicar las opciones que nos interesa.

Dentro del apartado Alcance tenemos dos posibles opciones: Público o Privado. La primera de ellas se utilizaría en el momento que deseamos crear un procedimiento que se pueda mirar desde cualquier formulario que tuviera una aplicación. Mientras que la segunda opción la utilizaríamos en el momento en el que queremos que el procedimiento sólo pueda ser consultado por el formulario en el que nos encontramos.

Dentro del apartado Tipo tenemos 4 opciones de las cuales sólo nos interesan

2: Procedimiento y Función. Vamos a ver que son cada una de estas opciones.

#### **Procedimiento o Sub:**

Un procedimiento ejecuta una tarea específica dentro de un programa sin devolver ningún valor.

#### **Función o Function:**

Una función ejecuta una tarea específica dentro de un programa, pero nos devuelve un valor.

En nuestra aplicación lo que nos interesa es un Procedimiento (Sub) ya que lo que deseamos es que se realicen una serie de instrucciones, pero no necesitamos que se nos devuelva ningún valor concreto.

36. Dentro de la ventana Agregar procedimiento escribe Crear en el apartado Nombre.  
37. Deja seleccionada la opción Procedimiento y escoge la opción Privado dentro del apartado Alcance.

Observa como dentro del código han aparecido estas líneas de código:

---

```
Private Sub Crear()
```

```
End Sub
```

---

Podrás ver que son muy parecidas a las líneas de código de los eventos de los diferentes elementos.

Dentro de estas dos nuevas líneas de código vamos a escribir el código que se repite dentro de nuestra aplicación.

---

```
Private Sub Crear()  
    For Contador = 1 To Elementos  
        Tabla(Contador) = Int((9 * Rnd) + 1)  
        Lista.AddItem Tabla(Contador)  
    Next Contador  
End Sub
```

---

Ahora ya estamos preparados para hacer que estas líneas se ejecuten en el momento en el que nosotros deseemos.

Para llamar a este procedimiento simplemente deberemos poner el nombre de este en el punto de la aplicación que deseemos. Por ejemplo vamos a ver como lo haríamos dentro del evento: **Form\_Load()**

Anteriormente este evento estaba creado de la siguiente forma:

---

```
Private Sub Form_Load()  
    Randomize  
    For Contador = 1 To Elementos  
        Tabla(Contador) = Int((9 * Rnd) + 1)  
        Lista.AddItem Tabla(Contador)  
    Next Contador  
End Sub
```

---

Si miramos las líneas que aparecen dentro del procedimiento que hemos creado anteriormente podremos ver donde deberemos hacer la llamada a **Crear**.

---

**Private Sub Form\_Load()  
Randomize  
Crear  
End Sub**

---

## Ejercicio 34 – Matrices de Controles

Vamos a realizar una practica en la que tendremos 5 botones, cada uno con una vocal. Nosotros lo que queremos es que cada vez que el usuario pulse un boton se acumulen las letras en un **Label**. Por ejemplo, si el usuario pulsa el boton con la **A**, despues la **O** y por ultimo el boton de la **E**. En el Label debera aparecer la cadena **AOE**.

Es un ejemplo sencillo, pero facil para comprender el funcionamiento de las matrices de controles.

1. Abre un nuevo proyecto
2. Inserta un LABEL, ponle como nombre CADENA, borra el contenido de dicha etiqueta
3. Inserta un boton. Cambia su nombre, poniendole LETRA.
4. Escribe en su interior la letra A.
5. Ajusta el tamaño del boton, el tamaño de la letra, su apariencia, etc. Realiza los cambios como desees. Ahora vamos a pasar a copiar ese elemento 4 veces mas para tener los demas botones, los cuales representaran al resto de las vocales.
6. Pulsa un click sobre el boton que tenemos en nuestro formulario para seleccionarlo.
7. Accede a la opcion COPIAR dentro del menu EDICION. Aparentemente no ha pasado nada. Pero la computadora ahora sabe que deseamos copiar este objeto.
8. Vuelve a acceder al menu EDICION, pero esta vez selecciona la opcion PEGAR. Acto seguido te aparecera un cuadro de dialogo en el que te avisa que ya existe un objeto que tiene el mismo nombre y si deseas crear una matriz de controles.
9. Contesta afirmativamente
10. Repite la accion de copiar hasta que tengas 5 botones.  
Las copias de los botones iran apareciendo en la esquina superior izquierda de nuestro formulario. Observa que en el momento que ya copias uno de los botones, contestanto afirmativamente a la creacion de una matriz de controles, el resto de copias las realiza sin hacerte ningun tipo de pregunta.
11. Situa uno debajo del otro todos los botones que hemos creado.
12. Selecciona el primero de los botones y observa la propiedad (Nombre)  
Podras observar como el nombre de dicho boton ya no es Letra, sino que pasa a ser Letra(0). Este numero, que aparece entre parentesis, es un indice el cual nos sirve para distinguir cada uno de los botones que hemos creado. Nuestra matriz de botones ira desde el boton con indice 0 al que tiene el indice 4. En total 5 elementos.

A partir de este momento, si queremos modificar la propiedad de uno de los botones en concreto, deberemos especificar el numero de indice de dicho boton. De esta forma podremos

utilizar estructuras de repeticion para modificar las propiedades de muchos objetos en pocas lineas. Mas adelante veremos ejemplos en los que utilizaremos esta caracteristicas.

Otra caracteristica que tienen las estructuras de datos es que todo el codigo que escribamos dentro de uno de ellos afectara igualmente a cualquiera de las copias.

13. Pulsa doble clic en cualquiera de los botones

Observa que en el momento de entrar en un evento, en la linea de codigo donde se especifica en que tipo de evento nos encontramos, aparecen las palabras **(Index As Integer)**. Este es el indice de los elementos que hemos insertado.

14. Escribe la siguiente instrucción:

---

**Cadena.Caption = Cadena.Caption & Letra(Index).Caption**

---

Esta instrucción lo que nos hace es “sumar” el contenido actual del Label llamado CADENA con el Caption del boton que hemos pulsado.

Antes de realizar una ejecucion de prueba deberemos hacer una ultima modificacion. Si te fijas todos los botones tienen como propiedad Caption una A. Vamos a modificar cada uno de los botones para que contengan las diferentes vocales.

15. Accede a cada uno de los botones y modifica la propiedad CAPTION escribiendo dentro de cada uno las diferentes vocales. No hace falta que esten en orden según el indice de cada boton.
16. Realiza una ejecucion de prueba
17. Pulsa los diferentes botones y observa como se van añadiendo las diferentes letras en nuestro LABEL.
18. Deten la ejecucion y graba nuestra pequeña practica.

## Ejercicio 35 – Ordenacion por burbuja

Elabore una aplicación que permita leer N números de tipo entero, y a continuación los visualice ordenados en forma ascendente o descendente.

Ordenación por burbuja

Ingrese un nuevo número:

14    Añadir

Orden:

☒ Ascendente    Ordenar

☐ Descendente

Lista de números:

2  
3  
14  
37  
68  
97

Salir

Para el desarrollo de esta aplicación, proceda a ubicar los siguientes controles en el formulario:

3 marcos

1 caja de texto

1 control lista

2 botones de opción

3 botones de comando

En seguida proceda a establecer las propiedades según se indica:

#### **Form1**

Nombre	FrmBurbuja
BorderStyle	3-Fixed Dialog
Caption	Ordenación por burbuja
Moveable	False

#### **Frame1**

Nombre	FraNumero
Caption	Ingrese un nuevo número:

#### **Frame2**

Nombre	FraLista
Caption	Lista de números:

#### **Frame3**

Nombre	FraOrden
Caption	Orden:

#### **Text1**

Nombre	TxtNumero
Text	

**List1**

Nombre	LstNumero
List	

**Option1**

Nombre	OptAscendente
Caption	Ascendente
Value	True

**Option2**

Nombre	OptDescendente
Caption	Descendente
Value	False

**Command1**

Nombre	CmdAnnadir
Caption	&Añadir
Default	True

**Command2**

Nombre	CmdOrdenar
Caption	&Ordenar

**Command3**

Nombre	CmdSalir
Caption	&Salir
Picture	C:\Archivos de programa\Microsoft Visual Studio\Common\Graphics\Icons\Arrows\Point04.ico

Style	1-Graphical
-------	-------------

Una vez establecidas las propiedades proceda a ingresar el código que se indica a continuación:

```

Private Sub CmdAceptar_Click()
    If IsNumeric(TxtNumero.Text) Then
        LstNumero.AddItem TxtNumero.Text
        TxtNumero.Text = ""
        TxtNumero.SetFocus
    Else
        MsgBox "Ingrese un número", vbCritical, "Mensaje"
        TxtNumero.SelStart = 0
        TxtNumero.SelLength = Len(TxtNumero.Text)
        TxtNumero.SetFocus
    End If
End Sub

Private Sub CmdOrdenar_Click()
    Dim I As Integer, J As Integer, T As Integer, N As Integer
    Dim A() As Integer
    N = LstNumero.ListCount
    ReDim A(N)
    For I = 0 To N - 1
        A(I) = LstNumero.List(I)
    Next I
    If OptAscendente.Value Then
        For I = 0 To N - 2
            For J = I + 1 To N - 1
                If A(I) > A(J) Then
                    T = A(I)
                    A(I) = A(J)
                    A(J) = T
                End If
            Next J
        Next I
    End If
End Sub

```



```

If OptDescendente.Value Then
  For I = 0 To N - 2
    For J = I + 1 To N - 1
      If A(I) < A(J) Then
        T = A(I)
        A(I) = A(J)
        A(J) = T
      End If
    Next J
  Next I
End If
LstNumero.Clear
For I = 0 To N - 1
  LstNumero.List(I) = A(I)
Next I
End Sub

Private Sub Form_Unload(Cancel As Integer)
  If MsgBox("Desea terminar la aplicación?", _
    vbQuestion + vbYesNo, "Pregunta") = vbYes Then

    End
  Else: Cancel = True : TxtNumero.SetFocus
End If
End Sub

Private Sub CmdSalir_Click()
  Unload Me
End Sub

```

## Ejercicio 36 – Pedido de cotizaciones

Elaborar una aplicación que permita seleccionar un artículo de un cuadro combinado (Combo). Apenas el usuario seleccione un artículo se debe mostrar el precio del mismo, el interés es fijo para esta ocasión.

El diseño de la interfaz debe ser similar a la siguiente figura:



La venta ha realizarse es a plazos, ello condiciona la cuota mensual a pagarse. Cuando se haga click sobre el botón Cuota mensual debe mostrarse un cuadro de diálogo con los datos propuestos:



De manera similar al hacer click sobre el botón Total nos debe mostrar la cantidad total a pagar.



Para el desarrollo de esta aplicación, proceda a ubicar los siguientes controles en el formulario:

- 1 marco
- 3 etiquetas
- 1 cuadro combinado
- 2 cajas de texto
- 3 botones de opción
- 2 botones de comando

En seguida proceda a establecer las propiedades según se indica:

#### Form1

Nombre	FrmCotizacion
BorderStyle	3-Fixed Dialog
Caption	Pedido de cotizaciones

#### Frame1

Nombre	FraPlazo
Caption	Plazo:

#### Label1

Nombre	LblArticulo
Caption	Artículo:

**Label2**

Nombre	LblPrecio
Caption	Precio US\$

**Label3**

Nombre	LblInteres
Caption	Interés:

**Combo1**

Nombre	CboArticulo
Text	

**Text1**

Nombre	TxtPrecio
Locked	True
Text	

**Text2**

Nombre	TxtInteres
Locked	True
Text	

**Option1**

Nombre	OptPlazo
Caption	6 meses
Value	True

**Option2**

Nombre	OptPlazo
Caption	12 meses

Value	False
-------	-------

### Option3

Nombre	OptPlazo
Caption	24 meses
Value	False

### Command1

Nombre	CmdCuotaMensual
Caption	&Cuota mensual

### Command2

Nombre	CmdTotal
Caption	&Total

Una vez establecidas las propiedades de la interfaz, haga doble click sobre el formulario e ingrese las siguientes declaraciones en la sección General del módulo de formulario:

#### Private Type Articulo

**Nombre As String \* 30**

**Precio As Double**

**End Type**

**Dim A(4) As Articulo, Plazo As Integer**

**Const Interes = 0.12**

Recuerde que un dato declarado en la sección General de un módulo puede ser accedido por todos los procedimientos de dicho módulo. Luego, continúe ingresando el código que se muestra a continuación:

#### Private Sub Form\_Load()

**A(0).Nombre = "Monitor SAMSUNG SyncMaster 3"**

**A(1).Nombre = "Impresora Hewlett Packard DeskJet 930C"**

**A(2).Nombre = "Impresora Epson Stylus Color 740"**

```

A(3).Nombre = "Microprocesador Pentium I 233 MHZ"
A(0).Precio = 150 : A(1).Precio = 275
A(2).Precio = 145 : A(3).Precio = 80
Dim I As Integer
For I = 1 To 4
    CboArticulo.AddItem A(I - 1).Nombre
Next I
TxtInteres = Interes : Plazo = 6
End Sub

Private Sub CboArticulo_Click()
    Dim I As Integer
    I = CboArticulo.ListIndex
    TxtPrecio = A(I).Precio
End Sub

Private Sub OptPlazo_Click(Index As Integer)
    Select Case Index
        Case 0: Plazo = 6
        Case 1: Plazo = 12
        Case 2: Plazo = 24
    End Select
End Sub

Private Sub CmdCuotaMensual_Click()
    Dim Total As Double, CuotaMensual As Double, I As Integer
    I = CboArticulo.ListIndex
    Total = A(I).Precio * (1 + Interes)
    CuotaMensual = Total / Plazo
    MsgBox "Cuota Mensual US$" & Str(CuotaMensual)
End Sub

Private Sub CmdTotal_Click()
    Dim Total As Double, I As Integer
    I = CboArticulo.ListIndex
    Total = A(I).Precio * (1 + Interes)
    MsgBox "Total US$" & Str(Total)
End Sub

```

## Ejercicio 37 – Ficha de matrícula

Se desea elaborar una aplicación que permita controlar el proceso de matrícula en un curso de computación. Para ello se deben recabar los siguientes datos: 1) Curso en que se matricula el alumno, 2) Fecha de matrícula, 3) Apellidos y nombres, 4) Sexo, 5) Dirección, y 6) Distrito de residencia.

**Ficha de matrícula**

**Ingreso de datos:**

Curso: Borland C++ Nivel I

Fecha de matrícula: 21/10/2000

Alumno: Meneses Correa, Juan Manuel

Sexo: Masculino

Dirección: Jr. Carlos Richardson N° 569

Distrito: Chorrillos

**Opciones:**

Guardar

Limpiar

Cancelar

Salir

**Cursos:**

- Borland C++ Nivel I
- Borland C++ Nivel II
- Microsoft Visual Basic Nivel I
- Microsoft Visual Basic Nivel II
- Microsoft Visual FoxPro Nivel I
- Microsoft Visual FoxPro Nivel II
- Microsoft Visual C++ Nivel I
- Microsoft Visual C++ Nivel II

**Distritos:**

- Barranco
- Breña
- Carabayllo
- Comas
- Chaclacayo
- Chorrillos
- El Agustino
- Jesús María

Para el desarrollo de esta aplicación proceda a ubicar los siguientes controles en el formulario:

- 4 marcos
- 6 etiquetas
- 5 cajas de texto
- 1 cuadro combinado
- 2 controles de lista
- 4 botones de comando

En seguida proceda a establecer las propiedades según se indica:

**Form1**

Nombre	FrmFichaMatricula
BorderStyle	3-Fixed Dialog
Caption	Ficha de matrícula

**Frame1**

Nombre	FraIngreso
Caption	Ingreso de datos:

**Frame2**

Nombre	FraOpciones
Caption	Opciones:

**Frame3**

Nombre	FraCursos
Caption	Cursos:

**Frame4**

Nombre	FraDistritos
Caption	Distritos:

**Label1**

Nombre	LblCurso
Caption	Curso:

**Label2**

Nombre	LblFechaMat
Caption	Fecha de matrícula:



**Label3**

Nombre	LblAlumno
Caption	Alumno:

**Label4**

Nombre	LblSexo
Caption	Sexo:

**Label5**

Nombre	LblDirección
Caption	Dirección:

**Label6**

Nombre	LblDistrito
Caption	Distrito:

**Combo1**

Nombre	CboSexo
Text	

**List1**

Nombre	LstCursos
Text	

**List2**

Nombre	LstDistrito
Text	

**Text1**

Nombre	TxtCurso
Text	

### Text2

Nombre	TxtFechaMat
Text	

### Text3

Nombre	TxtAlumno
Text	

### Text4

Nombre	TxtDireccion
Text	

### Text5

Nombre	TxtDistrito
Text	

### Command1

Nombre	CmdGuardar
Caption	&Guardar

### Command2

Nombre	CmdCancelar
Caption	&Cancelar

### Command3

Nombre	CmdLimpiar
Caption	&Limpiar

#### Command4

Nombre	CmdSalir
Caption	&Salir
Picture	C:\FundVB\Bitmaps\Exit.bmp
Style	1-Graphical

Una vez establecidas las propiedades de la interfaz, proceda a ingresar el código que se indica a continuación:

```
Private Sub Form_Load()  
    LstCursos.AddItem "Borland C++ Nivel I"  
    LstCursos.AddItem "Borland C++ Nivel II"  
    LstCursos.AddItem "Microsoft Visual Basic Nivel I"  
    LstCursos.AddItem "Microsoft Visual Basic Nivel II"  
    LstCursos.AddItem "Microsoft Visual FoxPro Nivel I"  
    LstCursos.AddItem "Microsoft Visual FoxPro Nivel II"  
    LstCursos.AddItem "Microsoft Visual C++ Nivel I"  
    LstCursos.AddItem "Microsoft Visual C++ Nivel II"  
    LstCursos.AddItem "Microsoft Visual J++ Nivel I"  
    LstCursos.AddItem "Microsoft Visual J++ Nivel II"  
    LstCursos.AddItem "Microsoft SQL Server Nivel I"  
    LstCursos.AddItem "Microsoft SQL Server Nivel II"  
    LstCursos.AddItem "Microsoft Power Builder Nivel I"  
    LstCursos.AddItem "Microsoft Power Builder Nivel II"  
    LstDistrito.AddItem "Callao"  
    LstDistrito.AddItem "Bellavista"  
    LstDistrito.AddItem "Carmen de la Legua"  
    LstDistrito.AddItem "La Perla"  
    LstDistrito.AddItem "La Punta"  
    LstDistrito.AddItem "Ventanilla"  
    LstDistrito.AddItem "Cercado de Lima"  
    LstDistrito.AddItem "Ancón"  
    LstDistrito.AddItem "Ate"  
    LstDistrito.AddItem "Barranco"  
    LstDistrito.AddItem "Breña"  
    LstDistrito.AddItem "Carabaylo"
```

**LstDistrito.AddItem “Comas”**  
**LstDistrito.AddItem “Chaclacayo”**  
**LstDistrito.AddItem “Chorrillos”**  
**LstDistrito.AddItem “El Agustino”**  
**LstDistrito.AddItem “Jesús María”**  
**LstDistrito.AddItem “La Molina”**  
**LstDistrito.AddItem “La Victoria”**  
**LstDistrito.AddItem “Lince”**  
**LstDistrito.AddItem “Lurigancho”**  
**LstDistrito.AddItem “Lurín”**  
**LstDistrito.AddItem “Magdalena del Mar”**  
**LstDistrito.AddItem “Miraflores”**  
**LstDistrito.AddItem “Pachacamac”**  
**LstDistrito.AddItem “Pucusana”**  
**LstDistrito.AddItem “Pueblo Libre”**  
**LstDistrito.AddItem “Puente Piedra”**  
**LstDistrito.AddItem “Punta Negra”**  
**LstDistrito.AddItem “Rimac”**  
**LstDistrito.AddItem “San Bartolo”**  
**LstDistrito.AddItem “San Isidro”**  
**LstDistrito.AddItem “Independencia”**  
**LstDistrito.AddItem “San Juan de Miraflores”**  
**LstDistrito.AddItem “San Luis”**  
**LstDistrito.AddItem “San Martín de Porres”**  
**LstDistrito.AddItem “San Miguel”**  
**LstDistrito.AddItem “Santiago de Surco”**  
**LstDistrito.AddItem “Villa María del Triunfo”**  
**LstDistrito.AddItem “San Juan de Lurigancho”**  
**LstDistrito.AddItem “Santa María del Mar”**  
**LstDistrito.AddItem “Santa Rosa”**  
**LstDistrito.AddItem “Los Olivos”**  
**LstDistrito.AddItem “Cieneguilla”**  
**LstDistrito.AddItem “San Borja”**  
**LstDistrito.AddItem “Villa el Salvador”**  
**LstDistrito.AddItem “Santa Anita”**  
**CboSexo.AddItem “Masculino” : CboSexo.AddItem “Femenino”**

**End Sub**

```

Private Sub Form_Unload(Cancel As Integer)
    If MsgBox("¿Desea terminar la aplicación?", _
        vbQuestion + vbYesNo, "Pregunta") = vbYes Then
        End
    Else
        Cancel = True
        Call CmdLimpiar_Click
    End If
End Sub

Private Sub LstCursos_Click()
    TxtCursos = LstCursos
End Sub

Private Sub LstDistrito_Click()
    TxtDistrito = LstDistrito
End Sub

Private Sub CmdGuardar_Click()
    LstCursos.Enabled = False
    LstDistrito.Enabled = False
    TxtCursos.Locked = True
    TxtFechaMat.Locked = True
    TxtAlumno.Locked = True
    CboSexo.Locked = True
    TxtDireccion.Locked = True
    TxtDistrito.Locked = True
    MsgBox "Alumno matriculado", vbInformation, "Mensaje"
End Sub

Private Sub CmdLimpiar_Click()
    LstCursos.Enabled = True
    LstDistrito.Enabled = True
    TxtCursos.Locked = False
    TxtFechaMat.Locked = False
    TxtAlumno.Locked = False
    CboSexo.Locked = False
    TxtDireccion.Locked = False

```

```

TxtDistrito.Locked = False
TxtCursos = ""
TxtFechaMat = ""
TxtAlumno = ""
CboSexo = ""
TxtDireccion = ""
TxtDistrito = ""
TxtCursos.SetFocus
End Sub

Private Sub CmdCancelar_Click()
    If MsgBox("¿Desea modificar algún dato?", _
                    vbQuestion + vbYesNo, "Mensaje") = vbYes Then
        LstCursos.Enabled = True
        LstDistrito.Enabled = True
        TxtCursos.Locked = False
        TxtFechaMat.Locked = False
        TxtAlumno.Locked = False
        CboSexo.Locked = False
        TxtDireccion.Locked = False
        TxtDistrito.Locked = False
    End If
End Sub

Private Sub CmdSalir_Click()
    Unload Me
End Sub

```

## PRACTICA 10 – Consulta de cursos

Desarrollar una aplicación que permita realizar consultas acerca de un determinado curso, los cuales se mostraran en una lista. El usuario debe seleccionar un curso y en seguida se debe presentar el nombre del profesor encargado del curso (teoría), el nombre del jefe de práctica (laboratorio), así como los horarios de teoría y de laboratorio. El diseño de la interfaz deberá ser similar al siguiente:

Consulta de cursos

Profesor de teoría: Castillo Peralta, Carlos

Horario de teoría: Sa 10-13

Jefe de práctica: Córdoba Saavedra, Javier

Horario de laboratorio: Do 08-10

Cursos:

- Microsoft Visual Basic Nivel II
- Microsoft Visual FoxPro Nivel I
- Microsoft Visual FoxPro Nivel II
- Microsoft Visual C++ Nivel I
- Microsoft Visual C++ Nivel II
- Microsoft Visual J++ Nivel I
- Microsoft Visual J++ Nivel II
- Microsoft SQL Server Nivel I
- Microsoft SQL Server Nivel II

## Ejercicio 38 – Concatenacion. Funciones de cadenas

1. Crea un nuevo formulario.
2. Inserta dos TextBox. Elimina el contenido y deja los nombres que tienen por defecto.
3. Inserta un Label. Elimina el contenido y ponle como nombre: Union.
4. Inserta un CommandButton. Pon el texto que quieras. No hace falta que cambies su nombre. Con este pequeño ejemplo queremos que al pulsar el botón, aparezca en el Label de nuestro formulario la concatenación del contenido de los dos TextBox.
5. Pulsa doble clic en el botón.
6. Escribe las siguientes líneas de código:

---

```
Private Sub Command1_Click()  
    Union.Caption = Text1.Text & Text2.Text  
End Sub
```

---

Recuerda que siempre que se realiza una asignación, pasa el contenido de la derecha del igual a la izquierda de este.

Para unir el contenido de los dos objetos insertados utilizamos el operador &.

7. Inicia una ejecución de prueba.
8. Escribe lo que quieras dentro de los dos TextBox.
9. Pulsa el Botón. Observa como el contenido de los dos TextBox pasa dentro del Label.
10. Cambia el contenido de los dos TextBox.
11. Pulsa nuevamente el Botón que hemos insertado en el formulario. Nuevamente vuelven ha aparecer los dos TextBox unidos dentro de nuestro Label.

¿Qué tendríamos que hacer para que entre ellos apareciera un espacio en blanco? La respuesta es muy sencilla, tendríamos que concatenar entre ellos un espacio en blanco.

12. Modifica la línea de código de la siguiente manera:

---

```
Private Sub Command1_Click()  
    Union.Caption = Text1.Text & " " & Text2.Text  
End Sub
```

---

Observa detenidamente la línea donde se produce la concatenación de los diferentes objetos. Primero concatenamos el contenido de Text1, después un espacio en blanco . . y por último el contenido de Text2.

13. Realiza una ejecución de prueba.
14. Detén la ejecución una vez hayas introducido diferentes textos en las correspondientes casillas. Si nosotros quisiéramos concatenar las casilla de texto separadas entre sí mediante una conjunción y deberíamos hacerlo de la siguiente manera:
15. Modifica las líneas de código para que queden así:

---

```
Private Sub Command1_Click()  
    Union.Caption = Text1.Text & " y " & Text2.Text  
End Sub
```

---

Observa que el texto que delante y detrás del texto que deseamos aparezca entre los elementos a concatenar hemos dejado un espacio, esto lo hacemos para que no salga junto a los elementos concatenados y la conjunción y. Observa igualmente que este texto está entre comillas.

16. Realiza una ejecución de prueba.



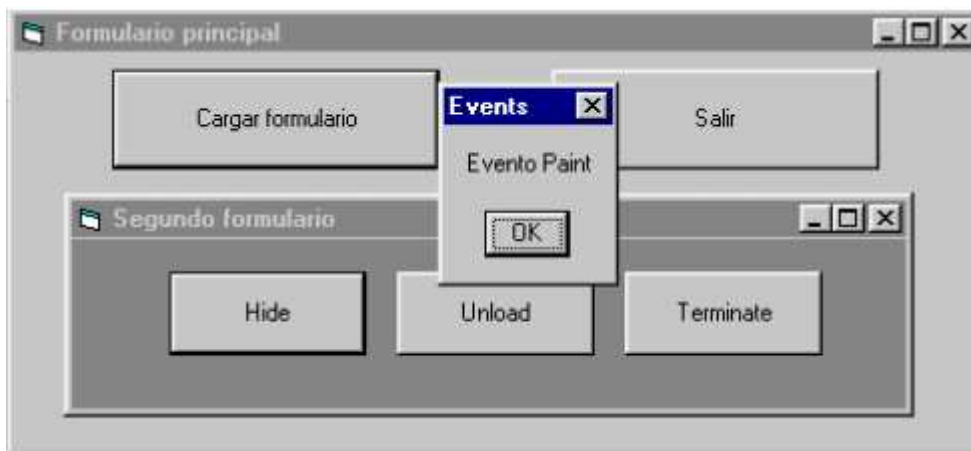
17. Detén la ejecución.

18. Modifica el código de nuestro ejemplo para cambiar el orden de la concatenación, escribiendo un punto posterior, el texto concatenado entre comillas, etc. Realiza todos los cambios que se te ocurran

19. Sal de Visual Basic sin guardar los cambios.

Recomendamos repasar con profundidad las estructuras de decisión (lección anterior) y las estructuras de repetición (lección actual).

## Ejercicio 39 – Eventos



Se han de crear dos formularios (frmPrincipal y frmSecundario). El primero de ellos contendrá dos botones (cmdVerSec y cmdSalir) y el segundo tres (cmdHide, cmdUnload y cmdTerminate).

El formulario principal será el primero que aparece, y solo se verá el segundo si se da clic en el botón CARGAR FORMULARIO. Cuando así se haga, a medida que los eventos antes mencionados se vayan sucediendo, irán apareciendo en pantalla unas cajas de mensajes que tendrán como texto el nombre del evento que se acaba de producir. Según con cual de los tres botones se haga desaparecer el segundo formulario, al volverlo a ver se produzcan unos eventos u otros.

### Código del programa

```
'codigo del form principal
Private Sub cmdCargar_Click()
    frmSecundario.Show
End Sub

'codigo del form secundario
Private Sub cmdHide_Click()
    Hide
End Sub
```

```
Private Sub cmdUnload_Click()  
    Unload Me  
End Sub  
  
Private Sub cmdTerminate_Click()  
    Hide  
    Set Form2 = Nothing  
End Sub  
  
Private Sub Form_Activate()  
    MsgBox("Evento Activate")  
End Sub  
  
Private Sub Form_Deactivate()  
    MsgBox("Evento Deactivate")  
End Sub  
  
Private Sub Form_Initialize()  
    MsgBox("Evento Initialize")  
End Sub  
  
Private Sub Form_Load()  
    MsgBox("Evento Load")  
End Sub  
  
Private Sub Form_Paint()  
    MsgBox("Evento Paint")  
End Sub  
  
Private Sub Form_QueryUnload( Cancel As Integer, UnloadMode As Integer)  
    MsgBox("Evento QueryUnload")  
End Sub  
  
Private Sub Form_Terminate()  
    MsgBox("Evento Terminate")  
End Sub  
  
Private Sub Form_Unload(Cancel As Integer)  
    MsgBox("Evento Unload")  
End Sub
```

## Ejercicio 40 – Menu

Antes de seguir trabajando vamos a explicar un poco en que consistirá nuestra aplicación de ejemplo.

Nosotros dispondremos de un formulario con solo dos objetos, un **TextBox** y un **ListBox**.

En el **ListBox** tendremos una serie de nombres ya escritos. Nosotros podremos añadir nuevos nombres en la lista utilizando el **TextBox**. Tendremos opciones en el **menú** para añadir el nombre a la lista y para borrar dicho nombre.

Con los elementos de la **lista** también trabajaremos, ya que podremos borrar alguno de los elementos o la lista completa, podremos bloquear la lista para que no se pueda ni borrar ni agregar elementos, y añadiremos una opción para cambiar el tamaño del texto de nuestra lista.

Todo esto utilizando solamente las opciones del menú.

Ahora que ya tenemos un poco claro de que va nuestra pequeña aplicación de ejemplo, vamos a empezar a crear la estructura de menús.

### ***Editor de menús***

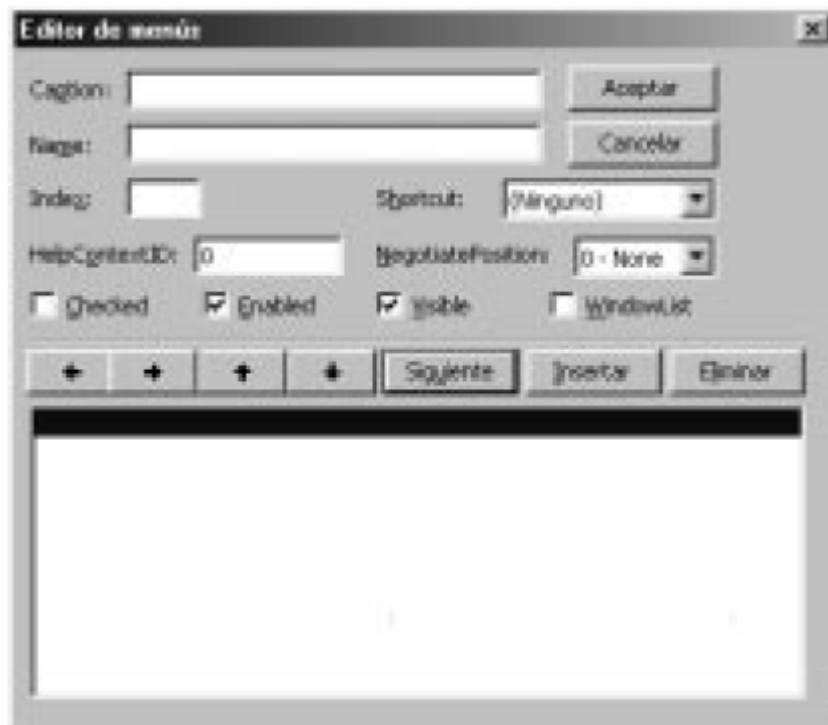
Para crear los diferentes menús que necesitaremos en una aplicación utilizaremos el **Editor** de menús. Esta herramienta nos permitirá crear toda la estructura de menús de forma sencilla.

1. Inicia **Visual Basic 6.0** con un formulario en blanco.
2. Accede a la opción **Editor** de menús dentro del menú **Herramientas**.



También puedes poner en funcionamiento el **Editor** de menús utilizando la combinación de teclas **[Control] + [E]** o utilizando en la **Barra** de herramientas estándar el siguiente botón:

Observa la nueva ventana que nos aparece en pantalla:



Vamos a comentar las principales partes de las que consta este **Editor** de menús. Las demás las iremos viendo conforme las necesitemos.

En los menús, como en la gran mayoría de objetos que forman parte de **Visual Basic**, las dos principales propiedades son el **Name** y el **Caption**. El **Name**, será el nombre que utilizaremos para hacer referencia al control del menú a lo largo de toda la aplicación y el **Caption** será el texto que aparecerá en el menú y que será por el cual se debe guiar el usuario. Piensa que el **Caption** debe ser corto y lo suficiente explicativo como para que el usuario entienda que es lo que pasa cuando se utiliza este control.

## ***Título de menú***

En nuestra aplicación vamos a necesitar dos menús diferentes. Uno que gestionará el **TextBox** y otro el **ListBox**.

Vamos a crear los dos **títulos** de menú.

El que gestionará el **TextBox** vamos a llamarle **Nombre** ya que aquí es donde escribiremos los nombres para insertarlos en la lista y al menú del **ListBox** le llamaremos **Lista**.

3. Sitúate sobre la casilla **Caption**.

#### 4. Escribe &Nombre

Recuerda que el símbolo & se utiliza para crear una tecla de acceso. En este caso la tecla de acceso al menú **Nombre** sería la **N** (Alt + N).

Observa que mientras escribes, la palabra **&Nombre** también aparece en el recuadro inferior de la ventana. A este cuadro le llamaremos **Cuadro de lista**. En este cuadro vamos a ir viendo una representación de las opciones que vamos insertando en nuestros menús.

#### 5. Pasa a la casilla **Name**.

Recuerda que aquí escribiremos el nombre con el que haremos referencia a este menú durante el código de la aplicación.

#### 6. Escribe **Nombre**. (Utiliza siempre nombres que te sean fáciles de recordar).

#### 7. Pulsa el botón **Siguiente**.

Observa como en el **cuadro de lista** la franja azul de selección pasa a la siguiente línea.

#### 8. Sitúate sobre el **Caption** y escribe **&Lista**.

#### 9. Ahora en el **Name** escribe **Lista**.

Ahora ya tenemos creados los dos **Títulos** de menú. Vamos a ver como quedan dentro de nuestro formulario.

#### 10. Pulsa el botón **Aceptar**.

Acto seguido estaremos de nuevo en el formulario de nuestra aplicación. Observa como han aparecido los dos títulos de menú que hemos creado anteriormente.



Antes de seguir trabajando con los menús vamos a colocar en nuestro formulario los dos objetos que necesitamos para llevar a cabo la aplicación.

#### 11. Sitúa donde quieras un **TextBox**.

12. Borra su contenido y ponle como **(Nombre): EntradaNombre**.
13. Sitúa donde quieras un **ListBox**.
14. Llámale **ListaNombres** y coloca en su interior 6 nombres de persona. Repasa lecciones anteriores.

## ***Interior de un menú***

Vamos a crear el contenido del menú **Nombre**.

15. Vuelve a abrir el **Editor** de menús. Utiliza el método que prefieras.
16. Sitúate sobre el **cuadro** de lista en la palabra **Lista**.
17. Pulsa el botón **Insertar**.

Observa como se ha creado un espacio en blanco entre **Nombre** y **Lista**. Aquí vamos a crear las opciones que irán dentro del menú **Nombre**.

Con los nombres que introduzcamos dentro de nuestro **TextBox** vamos a realizar tan solo dos posibles operaciones. La primera sería: pasar el contenido del **TextBox** a la lista y la segunda: borrar el contenido del **TextBox**. Para ello vamos a crear dos opciones dentro del menú **Nombre**, la primera será **Añadir** y la segunda **Borrar**.

18. Sitúate sobre la casilla **Caption** y escribe **&Añadir**.
19. Ponle como **Name: NombreAñadir**.

Vamos a tomar como costumbre poner nombres que nos ayuden a identificar rápidamente a donde pertenece esta opción. Con **NombreAñadir** podremos ver que **Añadir** está dentro de la opción **Nombre**. De esta forma también podremos hacer distinción entre la opción **Añadir** que esté dentro de **nombre** y otra opción a la que podremos poner el mismo **Caption** pero no el mismo **Name** en otro menú cualquiera.

Si observas el cuadro de lista, podrás ver que la opción **Añadir** esta a la misma altura que **Nombre** y que **Lista**, cosa que no nos interesa. A nosotros nos interesaría que **Añadir** esté dentro de la opción **Nombre**. Vamos a ver como podemos arreglar esto.

Observa en el **Editor** de menús que disponemos de 4 botones con flechas en su

interior. Vamos a ver para que se utilizan cada una de ellas.



Empezaremos a explicar de izquierda a derecha: la primera flecha sirve para bajar de nivel, la segunda para aumentar de nivel, la tercera para mover un menú a posiciones superiores y la cuarta para mover un menú a posiciones inferiores.

Vamos a ver estas opciones en funcionamiento.

### Aumentar un nivel

Vamos a hacer que la opción **Añadir** esté dentro del menú **Nombre**.

20. Haz un clic sobre el botón que tiene una flecha que apunta hacia la derecha.

Observa el **cuadro** de lista. Verás que en la opción **Añadir** han aparecido cuatro puntos a su derecha. Esto nos indica que **Añadir** ya forma parte de **Nombre**.



Vamos a ver como ha quedado nuestro menú en el formulario.

21. Acepta el Editor de menús.

22. Haz un clic sobre el menú **Nombre**. Dentro de él aparecerá la opción **Añadir**.

Vamos a colocar la segunda opción que debe estar dentro de **Nombre**.

23. Abre nuevamente el Editor de menús.

24. Colócate sobre **Lista**.

25. Pulsa en **Insertar**.

26. Sitúate en la casilla **Caption** y escribe **&Borrar**.

27. Como **Name** escribe: **NombreBorrar**

Como esta opción también debe ir dentro del menú **Nombre**, deberemos au-

mentar el nivel de la opción **Borrar**.

28. Pulsa sobre la flecha que apunta hacia la derecha.

29. Acepta el Editor de menús.

## ***Introducir código en los menús***

Como si se tratase de cualquier otro objeto, las diferentes opciones de nuestros menús también tienen eventos y también se puede escribir código en su interior.

30. Abre el menú **Nombre**.

31. Haz un clic sobre la opción **Borrar**.

Acto seguido te aparecerá la ventana del editor de código.

32. Escribe el siguiente código:

---

```
Private Sub NombreBorrar_Click()  
EntradaNombre.Text = «»  
End Sub
```

---

Con esto lo que conseguiremos es borrar el contenido del objeto **EntradaNombre**.

33. Cierra la ventana de código.

34. Haz un clic sobre la opción **Añadir** dentro del menú **Nombre**.

---

```
Private Sub NombreAñadir_Click()  
ListaNombres.AddItem (EntradaNombre.Text)  
End Sub
```

---

35. Escribe el siguiente código:

Con este código lo que conseguiremos es que el contenido del objeto **EntradaNombre** se añada a la **ListaNombres**.

Vamos a ver estas opciones en funcionamiento.



36. Realiza una ejecución de prueba.

37. Escribe cualquier nombre en **EntradaNombre**.

38. Abre el menú **Nombre** y escoge la opción **Añadir**.

Observa como el contenido de **EntradaNombre** pasa a formar parte de la lista.

39. Abre el menú **Nombre** y escoge la opción **Borrar**.

El texto que hay en **EntradaNombre** desaparece, se borra.

En un principio todo funciona bien, pero vamos a ver que ocurre en este caso:

40. Con la **EntradaNombre** vacía, selecciona la opción **Añadir**.

Aparentemente no ocurre nada.

41. Escribe un nombre en **EntradaNombre** y añádelo a la lista.

Como puedes observar, en el paso 40 lo que ha ocurrido es que hemos añadido un espacio en blanco a la lista, cosa que no nos interesa. Tendremos que pensar algo para que el usuario no introduzca elementos vacíos en la tabla.

## ***Activar y desactivar menús***

Vamos a ver como podemos activar y desactivar un menú cuando a nosotros nos interese. Esto siempre dependerá del “estado” en el que se encuentra la aplicación.

Vamos a desactivar todo el menú **Nombre** en el momento en el que **EntradaNombre** no contenga nada en su interior y vamos a activarlo nuevamente cuando el usuario escriba cualquier cosa.

42. Detén la ejecución del programa.

43. Pulsa doble clic sobre el objeto **EntradaNombre**.

De esta forma abriremos el evento **Change**. Evento que se pondrá en funcionamiento cada vez que se modifique el contenido de **EntradaNombre**.

44. Escribe el siguiente código:

---

```
Private Sub EntradaNombre_Change()  
    If Len(EntradaNombre.Text) <> 0 Then  
        Nombre.Enabled = True  
    Else  
        Nombre.Enabled = False  
    End If  
End Sub
```

---

Este código realiza lo siguiente: cuando se produce un cambio en **EntradaNombre** miramos el tamaño de este objeto. Esto lo haremos utilizando la instrucción **Len()**. Dentro de los paréntesis escribiremos el objeto al que queremos mirar el tamaño. Si el tamaño es **diferente** de 0, quiere decir que hay algo, entonces hacemos que el menú **Nombre** esté **activado**, mientras que si el tamaño es **igual** a 0 desactivamos el menú. De esta manera controlamos que el usuario no haga clic en este objeto.

En el momento que se inicia la ejecución el objeto **EntradaNombre** está **vacío**, pero el menú **Nombre** está **activado**. Esto es así porque todavía no se ha entrado en el evento **Change** del objeto **EntradaNombre** y no se han realizado las instrucciones que hemos escrito anteriormente. Vamos a ver que podemos hacer para que desde un principio este objeto esté **desactivado**.

45. Accede al **Editor** de menús.

46. Sitúate sobre la opción **Nombre**.



47. Busca esta opción dentro del **Editor** de menús:

48. Haz clic sobre ella.

49. Acepta el **Editor** de menús.

Observa como en nuestro formulario aparece la opción **Nombre** de color gris. En este momento ya no tenemos acceso a este objeto hasta que cambiemos la opción **Enabled**, ya sea desde el código o desde el **Editor** de menús.

Vamos a terminar de introducir las opciones que formarán parte del menú **Lista**.

50. Abre nuevamente el **Editor** de menús.

51. Sitúate en la siguiente línea de Lista.
52. Escribe en el **Caption**: &Borrar lista.
53. Escribe en **Name**: ListaBorrar.
54. Aumenta su nivel.
55. Pulsa en **siguiente**.

Observa como el siguiente objeto que insertemos ya tendrá el mismo nivel que Borrar Lista.

56. Escribe en el **Caption**: Borrar &elemento.
57. Escribe en **Name**: ListaBorrarElem.

Ahora ya tenemos dos objetos que forman parte del menú Lista.

Como ya hemos explicado al principio de esta misma lección en muchas ocasiones se utiliza una línea horizontal de separación para dividir opciones dentro de un mismo menú. Vamos a ver como podemos colocar nosotros una línea como esta dentro de nuestro menú.

### ***Líneas de separación***

58. Pulsa el botón **Siguiente**.
59. Escribe en el **Caption** un guión: -

Como cualquier otro objeto deberá tener nombre, aunque no podamos modificar sus propiedades.

60. Escribe en **Name**: ListaLinea
61. Acepta el **Editor** de menús.
62. Despliega el menú Lista y observa como en la última posición ha aparecido una línea horizontal que ocupa todo lo ancho del menú.
63. Accede nuevamente al **Editor** de menús.
64. Sitúate en la línea siguiente del último objeto.

65. Escribe en el **Caption**: **&Proteger**.

66. Escribe en **Name**: **ListaProteger**.

67. Pulsa en **siguiente**.

68. Escribe en el **Caption**: **-**.

69. Y como **Name**: **ListaLinea2**.

70. Pulsa en **siguiente**.

71. Escribe en el **Caption**: **&Tamaño**.

72. Escribe en **Name**: **ListaTamaño**.

Más adelante veremos para que utilizaremos las diferentes opciones que hemos puesto en nuestro menú e indicaremos el código que deberemos escribir dentro.

## ***Creación de submenús***

Vamos a crear un submenú dentro de la última opción que hemos insertado en el menú **Lista**.

73. Pulsa en **siguiente**.

74. Escribe en el **Caption**: **8**.

75. Y como **Name**: **ListaTamaño8**.

Ahora para que esta opción forme parte de un submenú de la opción **tamaño** deberemos aumentar el nivel, con lo que en la parte izquierda de este **8** aparecerán 8 puntos.

76. Aumenta de nivel esta opción.

77. Acepta el **editor** de menús.

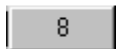
78. Sitúate sobre la opción **Lista**.

79. Despliega el menú.



Aparecerá un menú igual a este:

Observa como en la parte derecha de la opción **Tamaño** aparece una pequeña punta de flecha. Esto nos marca que en esta opción existe un submenú.



80. Sitúate sobre la opción **Tamaño** y observa lo que pasa.

A la derecha de la opción **Tamaño** ha aparecido el último elemento que insertamos a terminar de colocar los últimos elementos que forman parte de nuestros menús.

81. Abre nuevamente el **Editor** de menús.

82. Sitúate en la línea siguiente al 8.

83. Escribe en el **Caption:** 12.

84. Escribe en **Name:** ListaTamaño12.

85. Si es necesario aumenta su nivel hasta alcanzar el mismo que la opción anterior.

86. Pulsa en **Siguiente**.

87. Escribe en el **Caption:** 18.

88. Escribe en **Name:** ListaTamaño18.

Como te puedes imaginar vamos a utilizar estas tres opciones para cambiar el tamaño de letra de los objetos de la lista. Vamos a utilizar una nueva característica que nos brindan los menús, la **marca de verificación**.

## ***Marca de verificación***

Las marcas de verificación será una pequeña señal que aparecerá en la parte izquierda de una opción del menú. Esta marca nos servirá para saber si esta opción está o no activada. En nuestro caso en el momento que cambiemos el tamaño de la lista aparecerá una marca indicando cual es el tamaño actual de la lista.

Vamos a marcar una de las opciones inicialmente que será exactamente el tamaño de letra que tiene al iniciar la aplicación nuestra **Lista**.

Antes de realizar la marca vamos a mirar el tamaño de letra que tiene nuestra

89. Selecciona el objeto lista.

90. Accede al cuadro de diálogo **Fuente**. (Mira lecciones anteriores).

91. Si es necesario pon el tamaño de la fuente a **8**.

92. Acepta el cuadro de diálogo actual.

Ahora vamos a activar la opción **Tamaño – 8**, para indicar que este es el tamaño de fuente actual de la lista.

93. Selecciona el formulario.

94. Accede al **Editor** de menús.

95. Sitúate sobre el **8** del cuadro de lista y haz un clic sobre la opción **Checked**.

Aparentemente no ha ocurrido nada.

96. Acepta el Editor de menús.



97. Abre el menú **Lista**, dentro de él abre el submenú **Tamaño** y observa como al lado

del **8** aparece una marca como esta:

Vamos a ver como podemos activar y desactivar marcas de verificación utilizando el código.

98. Haz un clic sobre el **Tamaño – 12**.

99. Escribe el siguiente código:

---

```
Private Sub ListaTamaño12_Click()  
    ListaTamaño8.Checked = False  
    ListaTamaño12.Checked = True  
    ListaTamaño18.Checked = False  
    ListaNombres.FontSize = 12  
End Sub
```

---

Con este código estamos activando la opción **ListaTamaño12** y estamos desactivando las demás. Al desactivar las demás opciones nos ahorramos mirar de que opción venimos.

Después lo que hacemos es cambiar a **12** el tamaño de la letra de la lista.

100. Realiza una ejecución de prueba.

101. Cambia el **tamaño** de la lista a **12** utilizando las opciones del menú.

102. Observa como el objeto lista a cambiado de tamaño.

Esto es debido a que este objeto se adapta automáticamente para que ninguna de las líneas queden cortadas.

103. Detén la ejecución.

104. Selecciona el **ListBox**.

105. Accede a sus propiedades.

106. Cambia la propiedad **IntegralHeigth** a **False**.

A partir de este momento cuando cambiemos el tamaño de letra de la **Lista** esta no reducirá su tamaño.

Vamos a terminar de introducir el código para el cambio de tamaño.

107. Haz un clic sobre **Tamaño – 8**.

108. Escribe el siguiente código:

---

```
Private Sub ListaTamaño8_Click()  
    ListaTamaño8.Checked = True  
    ListaTamaño12.Checked = False  
    ListaTamaño18.Checked = False  
    ListaNombres.FontSize = 8  
End Sub
```

---

109. Cierra el editor de código.

110. Haz un clic sobre Tamaño – 18.

111. Escribe el siguiente código:

---

```
Private Sub ListaTamaño18_Click()  
    ListaTamaño8.Checked = False  
    ListaTamaño12.Checked = False  
    ListaTamaño18.Checked = True  
    ListaNombres.FontSize = 18  
End Sub
```

---

Observa el código de las tres opciones de tamaño. Ahora ya podemos cambiar el tamaño de letra de nuestra lista de nombres utilizando nuestro menú.

### ***Activar y desactivar Submenús***

Ahora vamos a pasar a escribir el código de la opción **Proteger**. Dentro de esta opción lo que queremos es que la lista de nombres quede bloqueada de tal forma que no se pueda hacer nada con ella. Para ello deberemos impedir que el usuario tenga acceso a alguna de las opciones de nuestro menú.

En este trozo de código volveremos a utilizar una partícula que ya vimos en lecciones anteriores (**Not**).

112. Accede a la opción **Proteger** dentro del menú **Lista** y escribe:



---

```
Private Sub ListaProteger_Click()  
    ListaNombres.Enabled = Not (ListaNombres.Enabled)  
    ListaProteger.Checked = Not (ListaProteger.Checked)  
    NombreAñadir.Enabled = Not (NombreAñadir.Enabled)  
    ListaBorrar.Enabled = Not (ListaBorrar.Enabled)  
    ListaBorrarElem.Enabled = Not (ListaBorrarElem.Enabled)  
    ListaTamaño.Enabled = Not (ListaTamaño.Enabled)  
End Sub
```

---

Observa cada una de las líneas e intenta averiguar para que se utilizan.

## ***Borrar lista***

Vamos a escribir el código para borrar el contenido de la lista. La primera línea de este código no se ha explicado, se hará en lecciones posteriores. Solo escríbela y en el momento de ejecutar la aplicación ya comprobarás para que sirve.

113. Accede a la opción Borrar lista y escribe el siguiente código:

---

```
Private Sub ListaBorrar_Click()  
    Respuesta = MsgBox(«¿Estás seguro?», 36, «Pregunta»)  
    If Respuesta = vbYes Then  
        ListaNombres.Clear  
        Lista.Enabled = False  
    End If  
End Sub
```

---

Con la instrucción `Clear` después del nombre de la lista, eliminamos el contenido de todos los elementos que forman parte de ella. También desactivamos la lista para que no se pueda trabajar con ella, ya que no contiene ningún elemento.

114. Haz una ejecución de prueba.

115. Borra el contenido de la lista.

Observa el mensaje que te aparece de confirmación.

116. Una vez borrada, escribe un nombre dentro de la casilla reservada para este efecto.

117. Añade el nombre mediante la opción del menú.

Observa como el menú Lista no se ha activado y nos interesa que lo hubiera hecho ya que ahora ya existen elementos en la lista para poder trabajar con ella.

Vamos a añadir una línea de código en una de las opciones que ya tenemos escritas.

118. Detén la ejecución y accede al código de Añadir del menú Nombre.

Recuerda que si el menú **Nombre** está desactivado no podrás entrar dentro de ninguna opción. Primero deberás activarlo utilizando el editor de menús.

119. Modifica el código para que quede de la siguiente forma:

---

```
Private Sub NombreAñadir_Click()  
  ListaNombres.AddItem (EntradaNombre.Text)  
  Lista.Enabled = True  
End Sub
```

---

La línea con el código en cursiva son las instrucciones que añadimos.

## ***Borrar elemento***

Vamos a ver como podemos eliminar un determinado elemento de la **lista** de nuestra aplicación.

120. Accede a Borrar elemento dentro del menú Lista.

121. Escribe el siguiente código que pasaremos a explicar a continuación:

---

```
Private Sub ListaBorrarElem_Click()  
  If ListaNombres.ListIndex = -1 Then  
    MsgBox "Debes seleccionar algún Elemento"  
  Else  
    ListaNombres.RemoveItem (ListaNombres.ListIndex)  
  End If  
End Sub
```

---

Antes de borrar algún elemento de la **lista**, nos vamos a asegurar que el usuario de la aplicación haya seleccionado algún nombre. Para ello utilizamos la instrucción **ListIndex** que nos devolvería el índice del elemento seleccionado. El índice, podríamos decir, que es la posición que ocupa el elemento seleccionado dentro de la lista. Es importante saber que el primer elemento de una lista tiene como índice **valor 0**.

Nosotros en la primera línea de este código preguntamos si **ListIndex** es igual a

—1, si el ordenador nos devuelve **verdadero** quiere decir que el usuario no ha seleccionado ningún elemento de la lista, con lo que mostraremos un mensaje de aviso. (Los mensajes los veremos en lecciones posteriores).

En cambio si **ListIndex** es diferente de —1 quiere decir que el usuario tiene seleccionado algún elemento con lo que ya podemos proceder al borrado. Esto lo haremos utilizando la instrucción **RemoveItem**. Entre paréntesis deberemos indicar el índice del elemento que ha seleccionado el usuario. La instrucción quedará de la siguiente forma: **ListaNombres.RemoveItem (ListaNombres.ListIndex)**.

Ahora ya podemos realizar ejecuciones de prueba para ver el funcionamiento de cada una de las opciones de nuestros menús.

Vamos a facilitar un poco el acceso a las diferentes opciones del menú, para ello utilizaremos las teclas de método abreviado.

## ***Teclas de método abreviado***

Las **teclas** de método abreviado nos permiten ejecutar las instrucciones de una opción determinada sin necesidad de desplegar ningún menú.

En nuestra aplicación vamos a utilizar esta propiedad en tan solo dos de las diferentes opciones: **Añadir Nombre** y **Proteger** la lista. No es conveniente abusar demasiado con las combinaciones de teclas ya que podemos “liar” al usuario.

122. Accede a **Editor** de menús.

123. Sitúate sobre la opción **Añadir**.

124. Busca dentro de la ventana **Editor** de menús la opción **Shortcut**.

125. Despliega la lista y busca dentro de toda esta lista, que representa las combinaciones de teclas de las que disponemos, **Ctrl+A**.

126. Pulsa un clic sobre ella y observa el **Cuadro** de lista.

Verás como en la parte derecha de esta opción aparecerá la combinación de teclas que hemos puesto.

127. Pulsa un clic sobre la opción **Proteger**.

128. Despliega la lista de **Shortcut** y selecciona **Ctrl+P**.

129. Acepta el **Editor** de menús.

Vamos a ver como funcionan estas nuevas propiedades.

130. Inicia una ejecución de prueba.

131. Sin desplegar ningún tipo de menú pulsa **Ctrl+P**.

Observa como la lista ha quedado protegida, de la misma forma que si hubiéramos accedido a la opción **Proteger** dentro del menú **Lista**.

Si abres el menú de la aplicación podrás ver como en el menú aparecen las combinaciones de teclas que hemos marcado para estas opciones.

132. Realiza todas las pruebas que desees, utilizando las opciones del menú.

Seguidamente vamos a introducir otro tipo de menú muy utilizado dentro de los programas creados para **Windows**. Los menús contextuales.

## ***Menú contextual***

Antes de trabajar con los menús contextuales vamos a ver que son y para que sirven

Un menú contextual aparece haciendo un clic con el botón derecho del ratón en alguna parte de la pantalla. Normalmente la gran mayoría de lugares de un programa contienen un menú contextual.

133. Detén la ejecución de la aplicación.

134. Sitúate sobre el formulario que estamos creando.

135. Haz clic con el botón derecho sobre él.

Seguidamente te aparecerá un nuevo menú con una serie de opciones. Estas opciones son las más utilizadas o las que nos puede interesar tener más a mano.

En nuestro ejemplo vamos a crear un menú contextual sobre la lista. De esta forma nos será mucho más fácil trabajar con las opciones que ya tenemos creadas dentro del menú **Lista**.

136. Haz doble clic en el objeto **ListaNombres**.

137. Selecciona dentro de la lista de procedimientos de este objeto: **MouseUp**.



Este evento se produce en el momento en el que el usuario suelta un botón del ratón.

Ahora nos interesaría introducir alguna instrucción que controlase si el usuario hace clic con el botón izquierdo del ratón o con el derecho. Recordemos que normalmente el menú contextual aparece haciendo clic con el botón derecho del ratón.

138. Escribe el siguiente código dentro del evento **MouseDown**.

---

**If Button = 2 Then PopupMenu Lista**

---

Vamos a comentar que es lo que hace el siguiente código.

Como ya hemos dicho anteriormente estas líneas de código se ejecutarán en el momento en el que el usuario pulsa y **suelta** un botón del ratón. Con la instrucción **If** miramos cual de los dos botones ha pulsado el usuario.

Cuando un usuario hace clic en uno de los botones del ratón, Visual Basic lo que hace es almacenar un valor en una variable llamada **Button**. Si el valor de esta variable es el **1** el usuario ha pulsado el botón izquierdo, mientras que si el valor devuelto es un **2** el usuario ha pulsado el botón derecho.

Para que se muestre el menú emergente o menú contextual utilizaremos la instrucción **PopupMenu** seguido del nombre del menú que deseamos mostrar. En nuestro caso el menú que queremos ver es el llamado **Lista**.

Con esta simple línea de código, situada dentro de este nuevo evento, conseguimos mostrar nuestro menú contextual. A partir de ahora en el momento en el que ejecutemos la aplicación, el botón izquierdo del ratón lo utilizaremos para seleccionar uno de los elementos de la lista, mientras que el botón derecho servirá para hacer aparecer nuestro menú contextual.

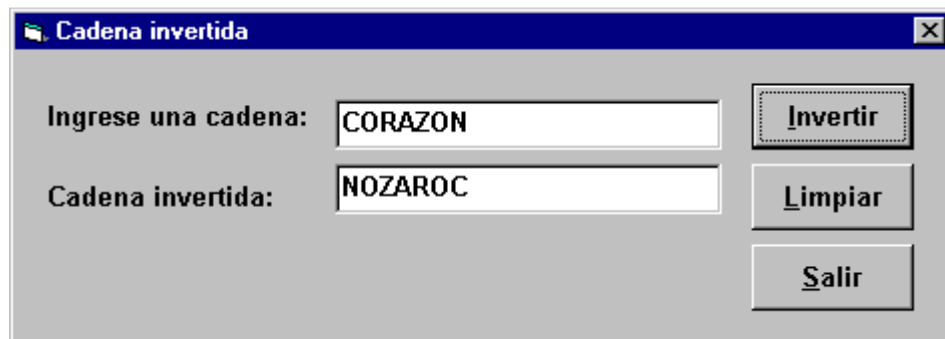
139. Realiza una ejecución de prueba y mira el funcionamiento de ambos botones dentro de nuestra lista.

Siempre que deseemos utilizar un menú contextual deberemos crearlo con el

Editor de menús. Si no deseamos que este aparezca en la barra de menús podremos ocultarlo.

## Ejercicio 41 – Cadena invertida

Escriba una función que reciba como argumento una cadena de caracteres y la devuelva en forma inversa, por ejemplo si se ingresa la cadena CORAZON deberá retornar NOZAROC.



Para el desarrollo de esta aplicación, proceda a ubicar los siguientes controles en el formulario:

2 etiquetas

2 cajas de texto

3 botones de comando

En seguida proceda a establecer las propiedades según se indica:

### Form1

Nombre	FrmCadInvertida
Caption	Cadena invertida
BorderStyle	3-Fixed Dialog

### Label1

Nombre	LblCadena
Autosize	True

Caption	Ingresa una cadena:
---------	---------------------

### Label2

Nombre	LblInvertida
Autosize	True
Caption	Cadena invertida:

### Text1

Nombre	TxtCadena
Text	

### Text2

Nombre	TxtInvertida
Locked	True
Text	

### Command1

Nombre	CmdAceptar
Caption	&Aceptar

### Command2

Nombre	CmdLimpiar
Caption	&Limpiar

### Command3

Nombre	CmdSalir
Caption	&Salir

Una vez establecidas las propiedades de la interfaz, proceda a ingresar el siguiente código:

```
Function CadInvertida(Cadena As String) As String
```

```
    Dim Invertida() As String * 1
```

```
    Dim I As Integer
```

```
    Dim J As Integer
```

```
    N = Len(Cadena)
```

```
    ReDim Invertida(N)
```

```
    For I = 1 To N
```

```
        Invertida(I - 1) = Mid(Cadena, I, 1)
```

```
    Next I
```

```
    For J = (N - 1) To 0 Step -1
```

```
        CadInvertida = CadInvertida & Invertida(J)
```

```
    Next J
```

```
End Function
```

```
Private Sub CmdInvertir_Click()
```

```
    TxtInvertida = CadInvertida(TxtCadena)
```

```
End Sub
```

```
Private Sub CmdLimpiar_Click()
```

```
    TxtCadena = ""
```

```
    TxtInvertida = ""
```

```
    TxtCadena.SetFocus
```

```
End Sub
```

```
Private Sub CmdSalir_Click()
```

```
    End
```

```
End Sub
```

## **Ejercicio 42 – Ficha de matricula (con varios forms)**

Se desea elaborar una aplicación que permita controlar el proceso de matrícula en un curso de computación. Para ello se deben recabar los siguientes datos: 1) Curso en que se matricula el alumno, 2) Fecha de matrícula, 3) Apellidos y nombres, 4) Sexo, 5) Dirección, y 6) Distrito de residencia. El diseño de la interfaz debe ser similar a la figura mostrada:



**Ficha de matrícula**

**Ingreso de datos:**

Curso: Borland C++ Nivel II ...

Fecha de matrícula: 21/10/2000

Alumno: Meneses Correa, Juan Manuel

Sexo: Masculino ▾

Dirección: Jr. Carlos Richardson N° 569


Distrito: Chorrillos ...

**Opciones:**

Guardar

Cancelar

Limpiar

 Salir

Para seleccionar un curso el usuario deberá hacer click en el botón punteado que se encuentra al lado de la caja de texto. En seguida se presentará un menú de selección por realce en el cual se presenta la relación de todos los cursos disponibles. El curso quedará seleccionado al hacer click en el botón Aceptar.

**Cursos**

Borland C++ Nivel I

**Borland C++ Nivel II**

Microsoft Visual Basic Nivel I

Microsoft Visual Basic Nivel II

Microsoft Visual FoxPro Nivel I

Microsoft Visual FoxPro Nivel II

Microsoft Visual C++ Nivel I

Microsoft Visual C++ Nivel II

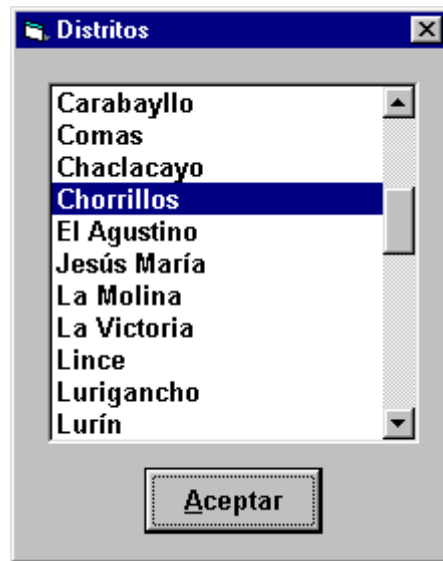
Microsoft Visual J++ Nivel I

Microsoft Visual J++ Nivel II

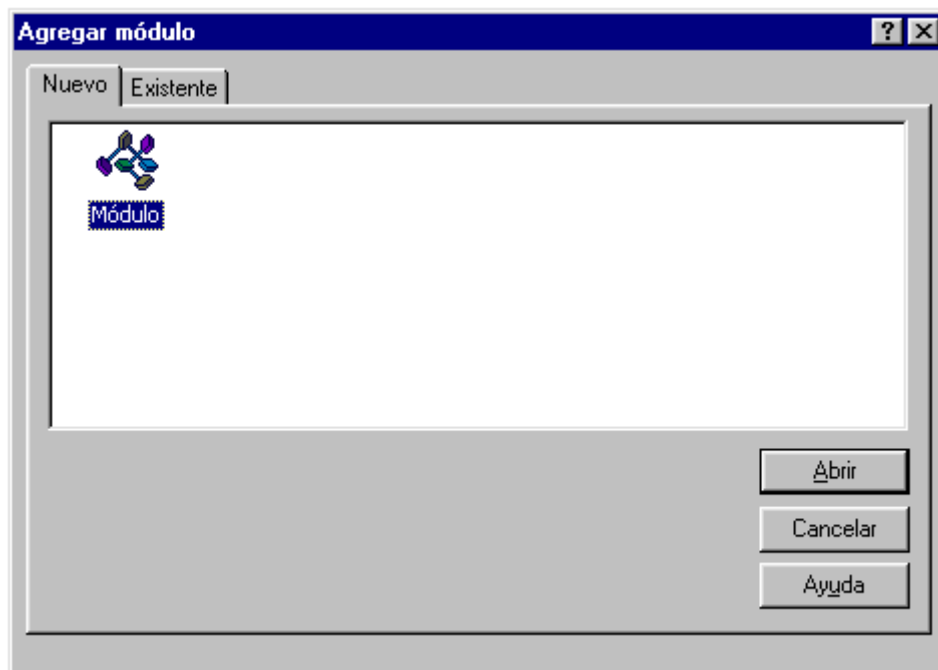
Microsoft SQL Server Nivel I

Aceptar

El mismo tipo de selección deberá realizarse al momento de ingresar el nombre del distrito.



Para el desarrollo de esta aplicación necesitamos tres formularios y un módulo. En primer lugar proceda a añadir un módulo de código al proyecto. Seleccione el Menú Proyecto y elija la opción Agregar módulo, se debe presentar un cuadro de diálogo similar a la siguiente figura:

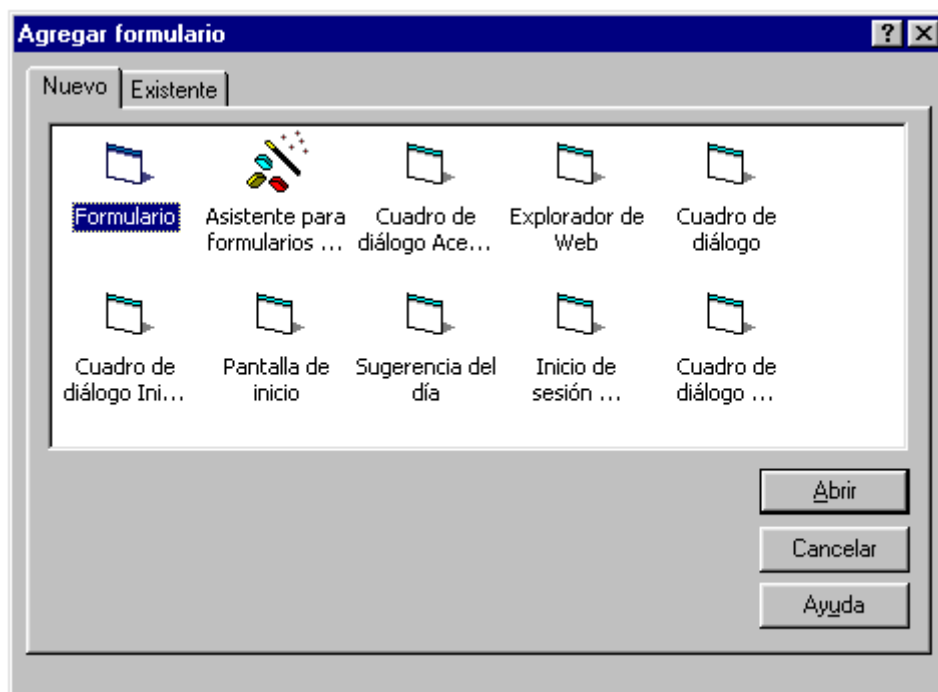


Del cuadro de diálogo Agregar módulo, en la ficha Nuevo, haga click en el botón Abrir. Luego ingrese el siguiente código en la sección de Declaraciones del módulo que acabamos de añadir:

## Public Curso As String

### Public Distrito As String

En seguida proceda a cambiar el nombre formulario principal por FrmFichaMatricula. Luego debe añadir los demás formularios necesarios para construir la aplicación. Para tal fin, seleccione el menú Proyecto y elija la opción Agregar formulario. Del cuadro de diálogo Agregar formulario, en la ficha Nuevo, elija la opción Formulario y haga click en el botón Abrir. Se debe presentar un cuadro de diálogo similar a la siguiente figura:



En ese instante se añadirá un nuevo formulario al proyecto. Cambie el nombre del nuevo formulario por FrmCurso. Repita el procedimiento anterior para añadir el formulario FrmDistrito.

A continuación copie los pasos de la pág. 55 a la pág. 62 de la Guía de Laboratorio Nº 4 (Aplicación Nº 3) con los siguientes cambios:

En la pág. 55, añadir sólo 2 marcos. No añadir ningún control de lista. En vez de 4 botones de comando, añadir 6 botones de comando. Luego, establecer las siguientes propiedades para los dos nuevos botones:

**Command5**

Nombre	CmdCurso
Caption	...

**Command6**

Nombre	CmdDistrito
Caption	...

En la pág. 59, reemplazar el código del evento Load del formulario por el siguiente:

**Private Sub Form\_Load()**

**CboSexo.AddItem "Masculino"**

**CboSexo.AddItem "Femenino"**

**End Sub**

En la pág. 61, suprimir el código asociado al evento Click de los controles de lista LstCursos y LstDistrito. En su lugar añadir lo siguiente:

**Private Sub CmdCurso\_Click()**

**Load FrmCurso**

**FrmCurso.Show vbModal**

**TxtCurso = Curso**

**End Sub****Private Sub CmdDistrito\_Click()**

**Load FrmDistrito**

**FrmDistrito.Show vbModal**

**TxtDistrito = Distrito**

**End Sub**

A continuación active el formulario FrmCurso, para ello haga click sobre el mismo. En seguida proceda a ubicar los siguientes controles en el formulario:

1 control de lista

1 botones de comando

Luego proceda a establecer las propiedades según se indica:

### Form2

Nombre	FrmCurso
BorderStyle	3-Fixed Dialog
Caption	Cursos

### List1

Nombre	LstCursos
Text	

### Command1

Nombre	CmdAceptar
Caption	&Aceptar

Una vez establecidas las propiedades proceda a ingresar el código que se indica a continuación:

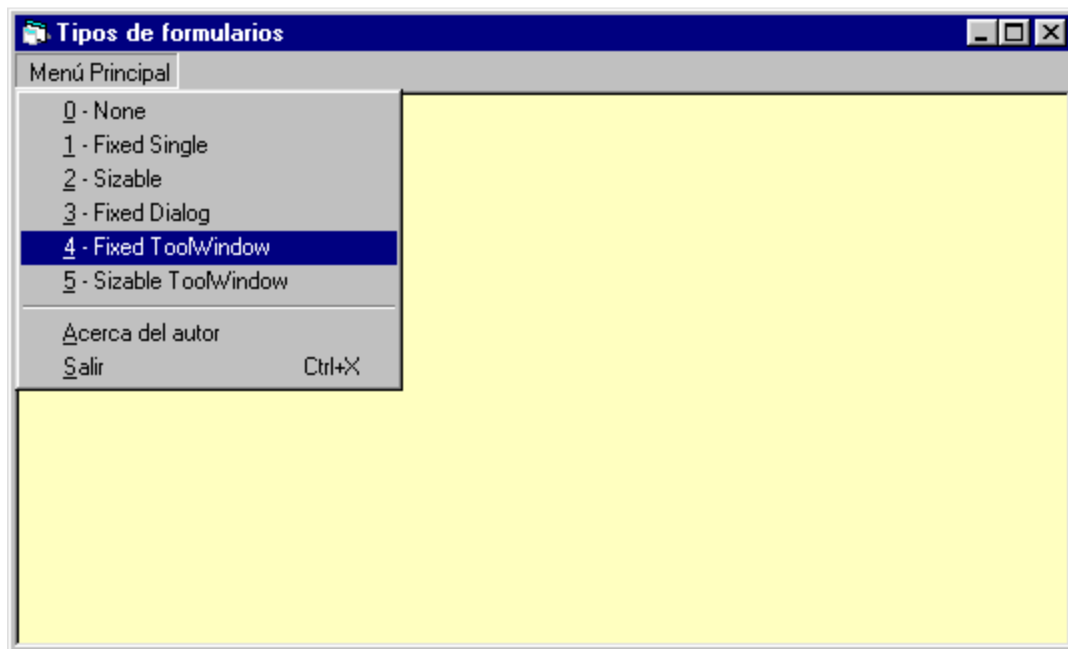
```
Private Sub Form_Load()  
    LstCursos.AddItem "Borland C++ Nivel I"  
    LstCursos.AddItem "Borland C++ Nivel II"  
    LstCursos.AddItem "Microsoft Visual Basic Nivel I"  
    LstCursos.AddItem "Microsoft Visual Basic Nivel II"  
    LstCursos.AddItem "Microsoft Visual FoxPro Nivel I"  
    LstCursos.AddItem "Microsoft Visual FoxPro Nivel II"  
    LstCursos.AddItem "Microsoft Visual C++ Nivel I"  
    LstCursos.AddItem "Microsoft Visual C++ Nivel II"  
    LstCursos.AddItem "Microsoft SQL Server Nivel I"  
    LstCursos.AddItem "Microsoft SQL Server Nivel II"  
    LstCursos.AddItem "Microsoft Power Builder Nivel I"  
    LstCursos.AddItem "Microsoft Power Builder Nivel II"  
End Sub  
Private Sub CmdAceptar_Click()
```

```
Curso = LstCursos.Text  
FrmCurso.Hide  
End Sub
```

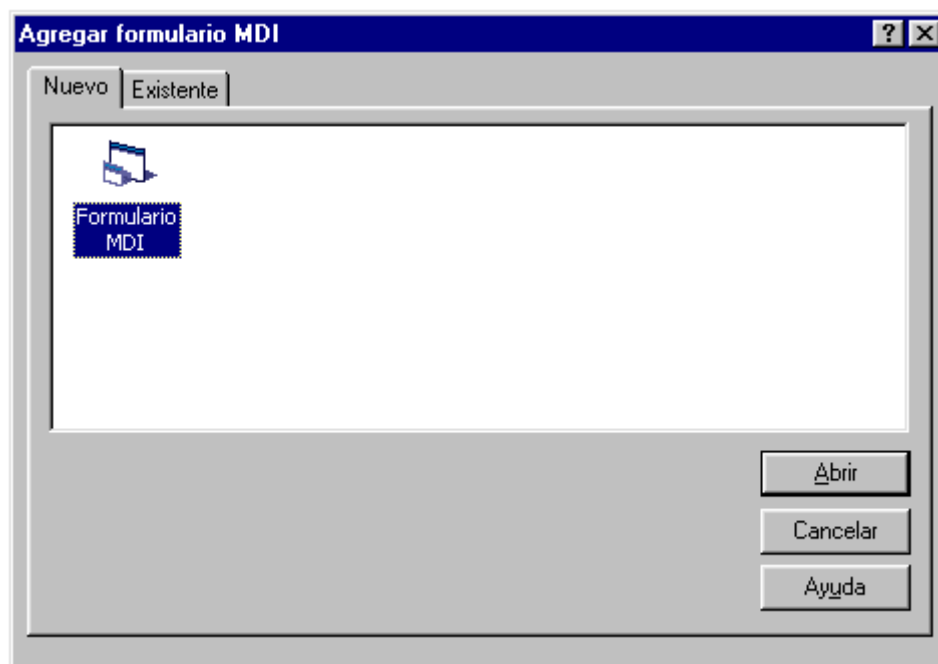
Por último, proceda Ud. a desarrollar el código respectivo para el formulario FrmDistrito.

## Ejercicio 43 – Tipos de Formularios (MDI)

Elaborar una aplicación que permita presentar los diferentes tipos de formularios de Visual Basic. Utilice como contenedor principal un formulario MDI, tal como se muestra en la figura:



Para el desarrollo de esta aplicación necesitamos utilizar un formulario MDI (interfaz de múltiples documentos). Para ello seleccione el Menú Proyecto y elija la opción Agregar formulario MDI, se debe presentar un cuadro de diálogo similar a la siguiente figura:



Del cuadro de diálogo Agregar formulario MDI, en la ficha Nuevo, haga click en el botón Abrir. En seguida cambie el nombre del formulario MDI por MDIPrincipal.

A continuación proceda a añadir los formularios para las diferentes opciones del menú. Cambie los nombres de los formularios según se indica:

Formulario	Nombre
Form1	FrmNone
Form2	FrmFixedSingle
Form3	FrmSizable
Form4	FrmFixedDialog
Form5	FrmFixedToolWindow
Form6	FrmSizableToolWindow
Form7	FrmAcercaDe

Luego proceda a diseñar el menú de opciones. Para ello haga click derecho sobre el formulario MDI y elija la opción Editor de menús. Establezca las propiedades según:

Caption	Name	ShortCut
&Menú Principal	MnuPrincipal	Ninguno
&0-None	MnuNone	Ninguno
&1-Fixed Single	MnuFixedSingle	Ninguno
&2-Sizable	MnuSizable	Ninguno
&3-Fixed Dialog	MnuFixedDialog	Ninguno
&4-Fixed ToolWindow	MnuFixedToolWindow	Ninguno
&5-Sizable ToolWindow	MnuSizableToolWindow	Ninguno
-	MnuLinea	Ninguno
&Acerca del autor	MnuAcercaDe	Ninguno
&Salir	MnuSalir	Ctrl + X

A continuación haga click en el botón Aceptar del Editor de menús. Luego proceda a ingresar el siguiente código para el formulario MDI:

```

Private Sub MDIForm_Unload(Cancel As Integer)
    If MsgBox("¿Desea terminar la aplicación?", _
        vbQuestion + vbYesNo, "Mensaje") = vbYes Then

        End
    Else: Cancel = True
    End If
End Sub

Private Sub MnuNone_Click()
    Load FrmNone
    FrmNone.Show
End Sub
Private Sub MnuFixedSingle_Click()
    Load FrmFixedSingle
    FrmFixedSingle.Show
End Sub

```



```

Private Sub MnuSizable_Click()
    Load FrmSizable
    FrmSizable.Show
End Sub

Private Sub MnuFixedDialog_Click()
    Load FrmFixedDialog
    FrmFixedDialog.Show
End Sub

Private Sub MnuFixedToolWindow_Click()
    Load FrmFixedToolWindow
    FrmFixedToolWindow.Show
End Sub

Private Sub MnuSizableToolWindow_Click()
    Load FrmSizableToolWindow
    FrmSizableToolWindow.Show
End Sub

Private Sub MnuAcercaDe_Click()
    Load FrmAcercaDe
    FrmAcercaDe.Show
End Sub

Private Sub MnuSalir_Click()
    Unload Me
End Sub

```

A continuación proceda a activar el formulario FrmNone y ubique un botón de comandos sobre el mismo. En seguida establezca las propiedades según se indica a continuación:

#### Form1

Nombre	FrmNone
BorderStyle	0-None
Caption	None
MDIChild	True

### Command1

Nombre	CmdVolver
Caption	&Volver

Una vez establecidas las propiedades, proceda a ingresar el código que se muestra a continuación:

```
Private Sub CmdVolver_Click()
```

```
    Unload Me
```

```
End Sub
```

Repita el procedimiento anterior para los demás tipos de formularios.

## PRACTICA 11 – Reserva de agua

Elaborar una aplicación que acepte como entrada la reserva de agua de un depósito y los litros que se consumen a la semana. Utilizando una función definida por el usuario determinar como resultado las cantidades de agua que quedan al final de cada semana. El proceso finalizará cuando no quede agua suficiente para una semana. Utilizar otro formulario para mostrar la salida.