MANUAL DE VISUAL BASIC I

Visual-Basic es una herramienta de diseño de aplicaciones para Windows, en la que estas se desarrollan en una gran parte a partir del diseño de una interface gráfica. En una aplicación Visual - Basic, el programa está formado por una parte de código puro, y otras partes asociadas a los objetos que forman la interface gráfica.

Es por tanto un termino medio entre la programación tradicional, formada por una sucesión lineal de código estructurado, y la programación orientada a objetos.

CONCEPTO DE PROYECTO.

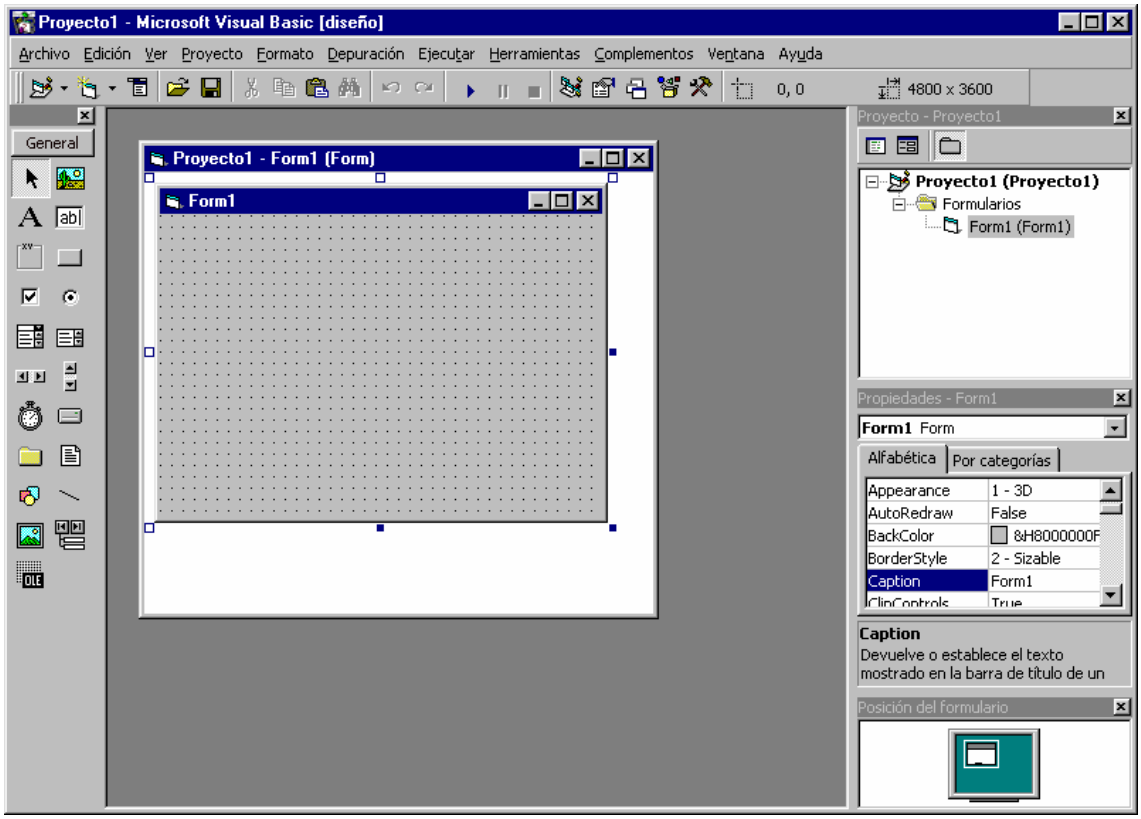
Dado que es muy común en aplicaciones Visual Basic compartir código o formularios personalizados, Visual Basic organiza las aplicaciones en lo que denomina proyectos. Cada proyecto puede tener varios formularios y, el código que activa los controles de un formulario es archivado con el formulario en archivos separados. El código general compartido por todos los formularios de una aplicación puede ser dividido en varios módulos, que también se archivan separadamente. En Visual Basic >5.0 un proyecto puede tener, además, módulos de clase y ficheros de recursos.
Aunque Visual Basic almacena separadamente los archivos que forman un proyecto, hace un seguimiento de dónde están los archivos. Crea un archivo con la extensión .VBP de Visual Basic Program/Project. Visual Basic permite tener un solo proyecto abierto en un momento determinado.
Los formularios se archivan con la extensión .FRM y contienen una imagen del formulario y, de todos los controles que pertenecen a él, incluidas sus propiedades. También pueden contener subrutinas de manejo de eventos, procedimientos generales, declaraciones de variables y de constantes a nivel de formulario y, procedimientos externos.
Un módulo estándar contiene código Visual Basic que no está asociado a ningún formulario en particular. Los procedimientos que se encuentran en el módulo, pueden ser accedidos desde cualquier otro procedimiento de la aplicación. Se almacenan con la extensión .BAS. Los módulos de clase tienen la extensión .CLS y contienen código, incluido subrutinas, funciones, métodos y procedimientos para crear descripciones genéricas de objetos. Estos módulos contienen propiedades que describen el comportamiento de una clase, así como el código que define las propiedades y los métodos de la clase.
Los archivos de recursos se almacenan con la extensión .RES y contienen bitmaps, cadenas de texto, o cualquier otra información que pueda ser cambiada sin tener que reeditar el código de la aplicación. Un proyecto solo puede contener un archivo de recursos.

INICIAR VISUAL BASIC:

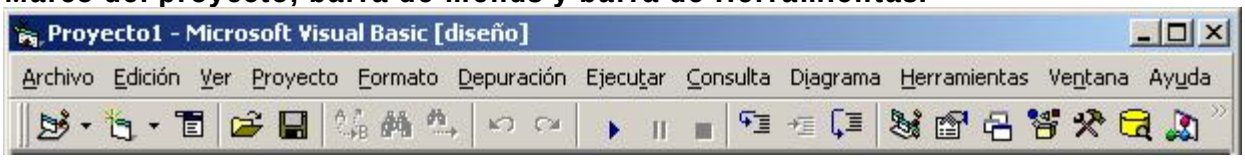
Al iniciar **Visual Basic** te aparecerá en primer termino una pantalla como esta:

Esta pantalla la comentaremos con mucho más detenimiento en próximas lecciones.

Observa la siguiente pantalla e identifica las partes:



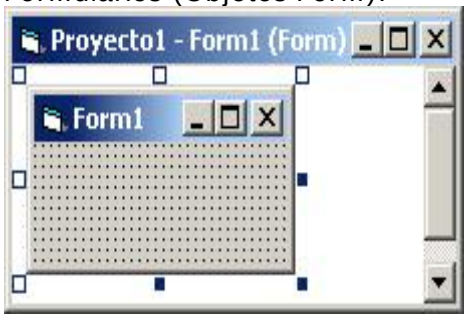
Marco del proyecto, barra de menús y barra de Herramientas.



En el marco superior de la ventana se puede apreciar el nombre del proyecto, y el formulario o módulo que estamos editando. En la barra de menús, están contenidas todas las opciones de que disponemos, tanto para la gestión del proyecto en general, como para la ventana de edición, el módulo u objeto que posea el foco en tiempo de diseño. En la barra de herramientas dispondremos de las opciones mas usadas de la barra de menús.

Ventana Editor del Proyecto.

Formularios (Objetos Form).



En esta ventana se editan los componentes con los que realizamos la aplicación, aqui se visualizan los formularios o sus ventanas de código, los módulos, módulos de clases, Informes de datos (DataReport).

En el caso de los formularios (Objetos Form), es la ventana que el usuario visualizará cuando ejecute la aplicación, aquí situaremos todos los controles para la iteración con el usuario. A través de sus propiedades el programador podrá modificar su entorno, haciendo que éste se pueda o no, maximizar o minimizar, mostrar una barra de título u ocultarla, especificar unas dimensiones fijas o variables, etc.. Sus métodos más importantes, son los de ocultarse o aparecer, cargarse en memoria o descargarse, ejecutándose los eventos implícitos del objeto FORM para estas acciones, también podemos desencadenar algunos de sus eventos mediante un click en uno de sus componentes, como son los botones de la barra de títulos, su marco, ó incluso detectando si el ratón está sobre su área de contenido, o si se ha dado una doble o simple pulsación sobre éste.

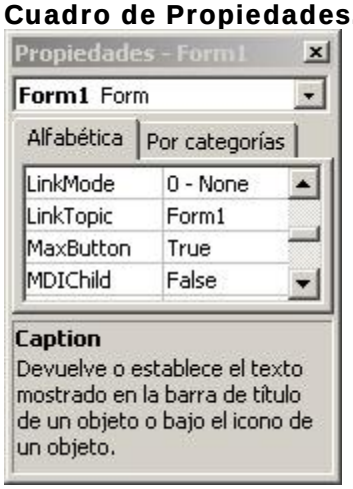
Ventana del Proyecto.

Aquí se listan, con su orden jerárquico, todos los formularios, módulos, módulos de clases, entornos de datos y demás componentes que conforman el proyecto o grupo de proyectos.

Al situar el foco sobre algún elemento de los aquí visualizados, si ese elemento tiene ventana de código, pulsando sobre el icono  (ventana de código), se abrirá el editor.

Si el elemento seleccionado posee ventana de diseño y la pulsación se realiza sobre el icono  (Objeto), se mostrará el formulario con los elementos y controles que tenga implementados.

Cuadro de Propiedades.



En este cuadro podemos asignar las propiedades del objeto, formulario o control que tenga el foco en tiempo de diseño.

Para poder cambiar una propiedad en varios controles a la vez, tiene que ir seleccionando éstos con el ratón mientras mantiene pulsada la tecla Ctrl. en ese caso solo se mostrarán las propiedades que sean comunes a todos los controles seleccionados.

Principio del formulario

Enviar

Final del formulario

Caja de Herramientas.

Aquí disponemos de las herramientas (Controles) necesarias para su inclusión en los formularios, Una vez abierto un proyecto nuevo, la caja de herramientas muestra unos controles por defecto, si el control que necesitamos no se encuentra, lo podemos añadir desde el menú 'Proyecto' en la opción 'Componentes', todas estas herramientas están disponibles en tiempo de diseño, a la vez que se pueden instanciar también durante la ejecución.

Objetos y controles más comunes:

FORM (Objeto Formulario): En el entorno de programación es la ventana donde situaremos todo o parte de la aplicación que estemos desarrollando y desde el lado del usuario será la ventana donde estarán contenidos todos los controles o información para su uso. A través de sus propiedades el programador podrá modificar su entorno, haciendo que éste se pueda o no, maximizar o minimizar, mostrar una barra de título o ocultarla, especificar unas dimensiones fijas o variables, etc.. Sus métodos más importantes, son los de ocultarse o aparecer, cargarse en memoria o descargarse, ejecutándose los eventos implícitos del objeto FORM para estas acciones, también podemos desencadenar algunos de sus eventos mediante un click en uno de sus componentes, como son los botones de la barra de títulos, su marco, ó incluso detectando si el ratón está sobre su área de contenido, e incluso si se ha dado una doble o simple pulsación sobre éste.

COMMANDBUTTON (Botón de comando): es uno de los controles más usado, observar que es muy común en cualquier formulario para ejecutar acciones como Aceptar, Aplicar, Salir, Cancelar o cualquier otra acción que necesite lanzar el usuario con su simple pulsación.

LABEL (Etiqueta): Nos muestra un texto dentro de un marco definido por sus propiedades, como pueden ser, fuente, color, tamaño, aspecto, color de fondo y apariencia.

TEXTBOX (Caja de texto): Como su nombre indica nos permite introducir caracteres dentro de un marco definido por sus propiedades al igual que el objeto LABEL. Posee además propiedades que nos permiten modificar el número de caracteres que éste puede contener.

CHECKBOX (Casilla de verificación): Muestra al usuario una casilla con un título, que al ser pulsado sobre ella presenta una marca que nos indica que su valor es activado (true) y si vuelve a pulsar esta marca, desaparece pasando su valor a desactivado (false).

FRAME (Cuadro Contenedor): Su imagen en el formulario es un recuadro que está definido por sus propiedades. Se usa para agrupar objetos y tratar conjuntamente éstos en el diseño gráfico y de comportamiento de la aplicación, es un objeto que al igual que el objeto FORM posee métodos comunes con los objetos que pueden ser contenedores de otros.

OPTIONBUTTON (Marcadores de opción): Muestra una casilla circular junto a un título, se usa para conmutar entre varias opciones, en el conjunto de los marcadores de opción que incluya el formulario no se puede seleccionar más de una opción a la vez, una técnica

común para poder incluir varios grupos de opciones, es hacer subconjuntos de éstos incluyéndolos en un control Frame.

 **LISTBOX (Cuadro de lista):** Dibuja un cuadro de texto donde cada línea es una opción seleccionable, es muy útil, cuando tenemos que elegir una opción en una lista de valores.

 **COMBOBOX (Cuadro de lista combinado):** Su aspecto en el formulario es el de un cuadro de texto con una flecha indicadora en su margen, que al ser pulsado despliega un cuadro igual que el ListBox, con una lista seleccionable, como su nombre indica combina un objeto TextBox con un ListBox.

 **IMAGE (Imagen):** Muestra una imagen especificada en su propiedad picture.

 **PICTUREBOX (Cuadro de imagen):** Al igual que el control IMAGE, con las mismas características que un objeto contenedor.

 **TIMER (Temporizador):** No es un control visible al usuario en tiempo de ejecución, mostrando un icono con forma de reloj en tiempo de diseño. Su propiedad específica es INTERVAL, donde se define la duración entre los intervalos del evento Timer. Mediante su propiedad Enabled, podemos controlar cuando queremos que empiece a ejecutar o detener su ciclo de intervalos.

 **DATA (Control de datos):** Enlaza un proveedor de datos con los consumidores de éstos, aportando los métodos necesarios para el tratamiento de los registros contenidos en la tabla enlazada, su principal evento es Validate que se ejecuta cuando se modifica algún registro y antes de su grabación ya sea desencadenado automáticamente por el desplazamiento del cursor de un registro a otro, o mediante código al ejecutar el método Update.
Existen muchos más controles, pero por ser un tema tan extenso, trataremos solamente los más utilizados, para mas información F1.

La ventana principal de V.B. encierra el entorno de desarrollo integrado (IDE). A continuación teneis una breve explicación de los elementos básicos que lo componen.

FORMULARIOS Y CONTROLES. PROPIEDADES COMUNES

PROPIEDADES MÁS COMUNES DE LOS FORMULARIOS:

AutoRedraw (Redibujado automático)

Gestiona la manera de redibujar las imágenes en pantalla. Puede tomar los valores True/False.

BackColor (Color de fondo)

Determina el color de fondo.

BorderStyle (Estilo del borde)

Solo se puede determinar en modo de diseño. Modifica el borde del formulario en el momento de ejecución.

Caption (Leyenda)

Establece el texto de la barra de título de la ventana del formulario. Esta propiedad indica lo que el usuario visualiza. No confundir con la propiedad Name. Propiedad común a muchos controles.

ControlBox (Cuadro de control)

Tiene efecto en tiempo de ejecución. Puede tomar los valores True/False.

Enabled (Habilitado)

El valor False hace que el formulario no responda a eventos.

Font (Fuente)

Determina el tipo de letra, atributos, etc. para impresión. El texto ya escrito no se verá afectado por un cambio en estas propiedades, pero sí el texto impreso posteriormente.

Height, Width (Ancho y Alto)

Determinan la anchura y altura del formulario. Se miden en Twips (1/20 punto o lo que es lo mismo, 567 twips en 1 centímetro) Determinan el tamaño del formulario cuando se imprima, no cuando sea visualizado en pantalla. A menos que el usuario modifique el borde

cambiando la propiedad BordeStyle, el usuario podrá modificar el tamaño y la forma de los diversos formularios de la aplicación sin tener en cuenta esta propiedad.

Icon (Icono)

Representa al formulario cuando esté minimizado o cuando se convierta en una aplicación independiente en el escritorio de Windows.

Left, Top (Izquierda, Arriba)

Determinan la posición del formulario dentro de la pantalla. Funcionan de manera idéntica a las propiedades Height y Width descritas anteriormente. Se miden en twips.

MaxButton, MinButton (Botones de maximizar y minimizar respectivamente)

El valor de esta propiedad (True/False) será ignorado si se ha establecido la propiedad BorderStyle a 0 - None. Solo es visible en tiempo de ejecución.

MousePointer (Puntero del ratón)

Determina la forma en que se mostrará el puntero del ratón.

Name (Nombre)

Propiedad MUY importante y común a todos los objetos de Visual Basic. Define el nombre del objeto en el código del programa. Para poder acceder a un control habrá que hacerlo a partir de su nombre. No disponible en tiempo de ejecución.

Picture (Dibujo)

Dibuja una imagen en el formulario (bitmap)

Visible

Determina si un formulario estará visible o no en tiempo de ejecución.

WindowState (Estado de la ventana)

Determina la forma en la que aparece el formulario durante la ejecución.

ScaleMode (Modo de la escala)

Permite cambiar las unidades de medida empleadas en el sistema de coordenadas interno del formulario.

ScaleHeight, ScaleWidth (Escala de la Altura y la Anchura)

Sirven para establecer una escala propia para la altura y anchura del formulario. Tiene efecto colateral sobre la propiedad ScaleMode, que se establece a 0.

ScaleLeft, ScaleRight (Escala de Izquierda y Derecha)

Sirven para establecer una escala propia para los márgenes izquierdo y superior del formulario. El valor original de estas propiedades es 0.

PROPIEDADES COMUNES DE LOS CONTROLES

Los controles que se pueden incluir en el formulario aparecen en la caja de herramientas. No es necesario recordar la posición o forma de cada control. En Visual Basic >5.0, cuando el cursor se sitúa sobre un control de la caja de herramientas, aparece una pista recordando el control que permite crear. Todos los controles tienen algunas propiedades comunes como pueden ser:

Caption (Leyenda)

Establece el texto que el usuario visualizará. No confundir con la propiedad Name.

Text (Texto)

Actúa igual que la propiedad Caption para aquellos controles que no dispongan de dicha propiedad, p.e. las cajas de texto. Muestra el contenido del control y, por tanto, también contiene los caracteres introducidos por el usuario. No confundir con la propiedad Name.

Name (Nombre)

Propiedad MUY importante. Define el nombre del control en el código del programa. No confundir con las propiedades Caption o Text que es lo que el usuario visualiza.

TabStop (Punto de Tabulación)

Si el valor es True, el control será susceptible de recibir el foco durante la ejecución de la aplicación.

TabIndex (Índice de tabulación)

Indica el número de orden en el que el control recibirá el foco cuando el usuario, en tiempo de ejecución, pulse la tecla Tab para recorrer los controles. A medida que se van situando controles en el formulario, Visual Basic incrementa en una unidad, el valor de esta propiedad para el nuevo control y, lo decrementa en caso de eliminar algún control. El valor para el primer control es 0.

PROPIEDADES DE LOS BOTONES DE COMANDO

Command Button (2ª fila, icono de la derecha). Se utiliza para ejecutar la acción asociada a la pulsación de dicho botón.

Enabled (Habilitado)

El valor False hace que el botón aparezca atenuado y, no responda a eventos.

Cancel (Cancelar)

Establecer el valor de esta propiedad a True, hace que el botón responda a la pulsación de la tecla ESC como si se hubiera hecho clic sobre él. En un formulario solo puede haber un botón con esta propiedad establecida a True.

Default (Defecto)

Establecer el valor de esta propiedad a True, hace que el botón responda a la pulsación de la tecla INTRO como si se hubiera hecho clic sobre él. Al igual como antes, en un formulario solo puede haber un botón con esta propiedad establecida a True.

PROPIEDADES DE LAS CAJAS DE TEXTO

Text Box (2ª fila, icono de la izquierda). Es un área dentro del formulario donde el usuario puede escribir texto o visualizarlo.

MaxLength (Tamaño máximo)

Determina el número máximo de caracteres que puede aceptar la caja de texto.

MultiLine (Líneas Múltiples)

Permite que la caja de texto admita varias líneas con la pulsación de la tecla INTRO. Normalmente se combina con la propiedad ScrollBars.

ScrollBars (Barras de desplazamiento)

Controla si en la caja de texto aparecerán las barras de desplazamiento o no.

PasswordChar (Carácter clave)

Permite mostrar un carácter clave en vez de los introducidos por el usuario.

Locked (Bloqueada)

Permite bloquear la caja de texto para que el usuario en tiempo de ejecución, no modifique el contenido de la caja. Esta propiedad es nueva en Visual Basic 5.0

SelLength (Longitud del texto seleccionado)

Número de caracteres seleccionado actualmente. Accesible durante la ejecución.

SelStart (Comienzo de la selección)

Indica dónde comienza el texto seleccionado (la posición del cursor). Si el valor es 0, el texto seleccionado empieza delante del primer carácter de la caja de texto. Si es igual a la longitud del texto de la caja, indica la posición detrás del último carácter del texto. Accesible en tiempo de ejecución.

SelText (Texto seleccionado)

Contiene el texto seleccionado. Accesible en tiempo de ejecución.

PROPIEDADES DE LAS ETIQUETAS

Label (1ª fila, icono de la derecha). Es un área dentro del formulario donde el usuario puede visualizar texto sin modificarlo.

Alignment (Alineación)

Determina la situación del texto dentro de la etiqueta.

AutoSize, WordWrap (Tamaño automático y enlace de textos)

La primera propiedad permite que la etiqueta crezca horizontalmente en función de su contenido. La segunda propiedad permite que el crecimiento sea vertical.

BOTONES DE OPCIÓN, CUADROS DE VERIFICACIÓN Y BARRAS DE DESPLAZAMIENTO

Los botones de opción (Option Button) y las casillas de verificación (CheckBox), indican un estado y permiten al usuario que cambie el estado. Las casillas de verificación actúan independientemente. Por el contrario, los botones de opción aparecen en conjuntos y permiten escoger un valor del conjunto.

Las propiedades importantes de estos controles son, prácticamente las mismas que las de un botón de comando, a excepción de la propiedad Value (valor), que indica el estado del control. Para los botones de opción puede estar establecida a True o False. Si está a True, el botón estará activado. Para las casillas de verificación, se podrá establecer a 0-Unchecked (no está activado), 1-Checked (activado) o 2-Grayed (atenuado).

MARCOS:

Los marcos (Frame) sirven para separar grupos de otros objetos en la pantalla. Para los controles anteriores, también afectan al comportamiento de ellos. Para asegurarse que los controles que se añadan al marco pertenecen al marco, se deben crear dentro del marco y no arrastrarlos encima del marco.

PROPIEDADES DE LOS CUADROS DE LISTA

List Box (4ª fila, icono izquierdo) Permite ofrecer al usuario una serie de opciones para que elija. Visual Basic añadirá barras de desplazamiento al cuadro de lista si la lista completa es demasiado larga para ser vista toda a la vez. El contenido de un cuadro de lista no se puede definir durante el diseño. En su lugar se utilizará el método AddItem para introducir elementos en la lista.

List (lista)

Esta propiedad no se puede modificar en fase de diseño, contiene la matriz de todos los valores almacenados en el cuadro de lista. Para acceder a un elemento de la lista se seguirá la sintaxis: Objeto.List(índice).

ListCount (contar lista)

No puede modificarse directamente. Contiene el número de elementos del cuadro de lista.

ListIndex (índice de la lista)

Indica el número de la lista más recientemente seleccionado. El valor del primer elemento es 0, el del segundo, 1, y así sucesivamente. Si no hay ningún elemento seleccionado, la propiedad tendrá un valor de -1.

Sorted (ordenada)

Para mantener la lista ordenada alfabéticamente.

Text (texto)

No se puede modificar directamente, contiene el texto del elemento más recientemente seleccionado.

Los métodos utilizados con los cuadros de lista son: AddItem (añadir elemento) permite insertar una línea de texto en el cuadro de lista. Su sintaxis es objeto.AddItem texto[, índice] . Clear (borrar) sirve para eliminar todos los elementos del cuadro de lista. Su sintaxis es: objeto.Clear. RemoveItem (eliminar elemento) permite eliminar una línea del cuadro de lista. Su sintaxis es: objeto.RemoveItem índice.

PROPIEDADES DE LOS CUADROS COMBINADOS

Combo Box (3ª fila, icono derecho) Puede combinar un cuadro de texto y un cuadro de lista en un solo control.

Style (estilo)

Determina el tipo de cuadro combinado y cómo se comporta. Puede tomar los siguientes valores: 0 - Dropdown Combo; 1 - Simple Combo; 2 - Dropdown List.

Text (texto)

Contiene el texto del elemento seleccionado o introducido por el usuario en el área de edición.

Los procedimientos descritos en el apartado anterior sirven para este tipo de control. Ambos tipos de control permiten asociar a los elementos del listado un número entero. Para ello se debe utilizar la matriz ItemData, que es una propiedad de estos controles. Para añadir los datos numéricos a esta matriz, se debe conocer el índice matriz del elemento, para lo que se utilizará la propiedad NewIndex del control. Es decir, se necesitará escribir código parecido a:

```
Listado1.AddItem Elemento_A_Añadir
Listado1.ItemData(Listado1.NewIndex) = Datos_Del_Entero
```

BARRAS DE DESPLAZAMIENTO

(HScrollBar y VScrollBar, 4ª fila, iconos central y derecho) Informan de la posición del cuadro de desplazamiento dentro de la barra.

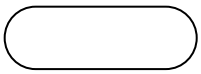
Las propiedades que se suelen utilizar son: Value (valor) contiene el número que representa la posición actual del cuadro de desplazamiento en el interior de la barra. LargeChange (gran cambio) representa el valor añadido o sustraído al número contenido en la propiedad Value

cuando el usuario hace clic dentro de la barra de desplazamiento. Max (máximo) indica el valor de la propiedad Value cuando el cuadro de desplazamiento está en su posición más a la derecha o abajo posible. Min (mínimo) indica el valor de la propiedad Value cuando el cuadro de desplazamiento está en su posición más a la izquierda o arriba posible. SmallChange (cambio pequeño) representa el valor añadido o sustraído al número contenido en la propiedad Value cuando el usuario hace clic dentro de una de las flechas de desplazamiento.
Los eventos asociados a este tipo de control son los eventos Change producido después de que se haya modificado la posición del cuadro de desplazamiento; y el evento Scroll, emitido repetidamente mientras el cuadro de desplazamiento se arrastra por el interior de la barra de desplazamiento (no ocurre si se mueve haciendo clic en las flechas o en la barra). Se utiliza para proporcionar información instantánea.

DIAGRAMAS DE FLUJO:

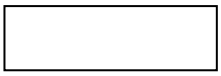
Un diagrama de flujo es una de las técnicas de representación de algoritmos mas utilizada desde la aparición de los lenguajes de programación estructurados.
Un algoritmo consiste en describir paso a paso con un lenguaje natural, en forma precisa, definida y debe tener un número finito de pasos.
Un diagrama de flujo utiliza símbolos estándar mostradas y tiene los pasos del algoritmos escritos en los símbolos unidos por flechas, denominadas líneas de flujo, que indican la secuencia en que se debe ejecutar.

Terminador



Representa el comienzo y el final de un programa. Puede representar también una parada o interrupción programada que sea necesario realizar en un programa.

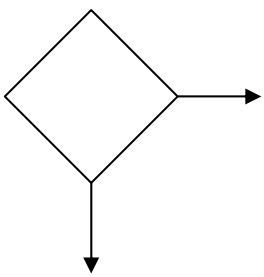
Proceso



Viene a ser cualquier tipo de operación que pueda originar cambio de valor, formato o posición de la información almacenada en memoria, operaciones aritméticas, de transferencia, etc.

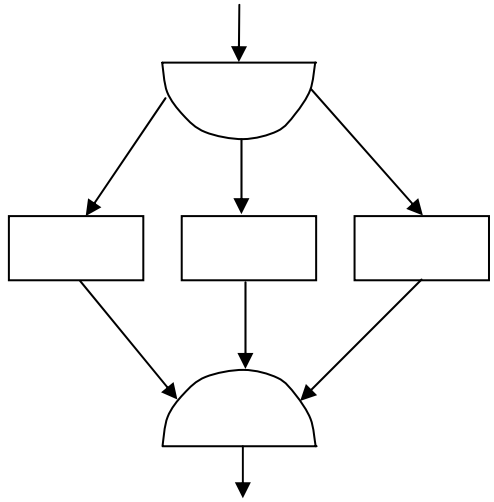
Datos (entrada/salida)
Entrada:
Representa cualquier introducción de datos en memoria desde los periféricos.
Salida:
Representa cualquier salida de información procesada en un periférico.

Decisión



Indica operaciones lógicas o de comparación entre dato, y en función al resultado de la misma determina cuál de los distintos caminos alternativos del programa se debe seguir.

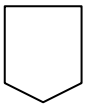
Decisión Múltiple



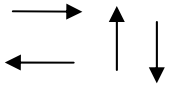
Su función es seleccionar los caminos alternativos.
Conectores

Conexión en la misma página:
Sirven para enlazar dos partes cualesquiera de un diagrama a través de un conector en la salida y otro conector en la entrada.

Conexión para páginas diferentes:
Es la conexión de dos puntos del diagrama situado en páginas diferentes.



Indicador de Dirección o línea de flujo:
Indica el sentido de ejecución de las operaciones.



Línea Conectora:
Sirve de unión entre dos símbolos

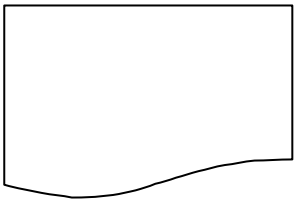
Preparación:
En este símbolo se describe la preparación de datos.

Llamada a subrutina o Proceso predefinido

Una subrutina es un módulo independiente del programa principal, que recibe una entrada procedente de dicho programa, realiza una tarea determinada y regresa, al programa principal.

pantalla

Impresora



VARIABLES y CONSTANTES

No podemos decir que estos elementos sean estructuras básicas, ya que no son un grupo de instrucciones, sino que son elementos que nos pueden ayudar a almacenar valores, de forma temporal, para usarlos en nuestra aplicación de la forma que más nos convenga.

Variables

Las variables se utilizan, en cualquier lenguaje de programación, para almacenar valores de forma temporal (mientras dura la ejecución del programa).
Definen una dirección de memoria donde se almacenarán valores para su posterior utilización, reconocibles en el código por el nombre con el que se declaran, pudiéndose recuperar el valor que contienen.
A las variables se les pone un nombre para poder trabajar con ellas y se indican de que tipo son. Este tipo nos informa que tipo de datos se pueden almacenar dentro de esta variables, (los diferentes tipos de variables los veremos más adelante).

Imagina que tienes que calcular el total de ventas de un mismo articulo, para tres puntos de venta distintos, el código a escribir sería algo parecido a esto:

VarTotalVentas = VarDepVent_1 + VarDepVent_2 + VarDepVent_3

En la variable VarDepVent_n, tendríamos guardados los valores por punto de venta y guardaríamos el total, de las sumas de esas ventas, en otra variable llamada VarTotalVentas

Constantes

Las constantes nos pueden parecer que son exactamente iguales que las variables, pero no es así. Las variables nos sirven para almacenar valores, que podemos modificar durante la ejecución del programa.

Las constantes, en cambio, no cambian de valor durante la ejecución de la aplicación. Se suelen utilizar para sustituir un número o valor, difícil de recordar o que suele salir muchas veces durante la aplicación. Imagina el caso de una aplicación en la que necesites utilizar muchas veces el valor Pi. Si siempre que necesitáramos este valor tuviéramos que escribir 3'1415... sería un poco engorroso. Pero, gracias a las constantes, nosotros podemos definir una llamada Pi con valor 3'1415... y en todo momento que necesitemos realizar una operación con el valor Pi solo tendremos que poner el nombre de la constante y la computadora se encargará de sustituirlo por su contenido.

Definición de la constante sin indicar el tipo de dato:
Const [Nombre Constante] = [Valor Constante]

Definición de la constante indicando el tipo de dato:
Const [Nombre Constante] As [Tipo de dato] = [Valor Constante]

Vida de una variable

Nosotros podemos definir una variable para que solo nos sea útil mientras dura el procedimiento en el que se le ha creado. En el momento de finalizar dicho procedimiento, el espacio reservado en memoria para la variable queda liberado. De esta forma, si nosotros queremos utilizar un mismo nombre para diferentes variables que se utilizan en diferentes procedimientos podemos hacerlo sin miedo a que los valores se mezclen o cambien sin que nosotros tengamos un control de dichos cambios.

Definir una variable

Para definir una variable dentro de un procedimiento utilizaremos la instrucción: **Dim [Nombre Variable] As [Tipo variable].**

Nombre Variable: definiremos el nombre que tiene la variable. Este nombre no puede tener más de 255 caracteres, debe comenzar con una letra y no debe contener puntos.

Tipo variables: Especificaremos el tipo de datos que se pueden almacenar dentro de la variable.
Tipos de datos:
Numéricos: Byte, Integer (entero corto), Long (entero largo), Single, Double, Currency (decimales).
Si una variable va a contener un número decimal, declararla como Single, Double o Currency. El tipo de dato Currency acepta hasta cuatro dígitos a la derecha del separador decimal y hasta quince dígitos a la izquierda; es un tipo de dato de signo fijo adecuado para cálculos en moneda. Las variables de signo flotante (Single y Double) tienen un mayor intervalo que Currency, pero pueden ocurrir pequeños errores de redondeo.
Cadena: String, este tipo almacena cadenas de caracteres. Para definir un tipo de cadena fija, basta con designar su longitud en la declaración.
Dim MiVar As String * 10
Fecha / Hora: Las variables tipo Date presentan fechas de acuerdo al formato de fecha corto reconocido por su sistema. La hora se presenta de acuerdo al formato de hora reconocido por su sistema (12 ó 24 horas).

Asignar valores

Variables

Para la asignación de valores tenemos que pensar, siempre, el tipo de datos que podemos almacenar dentro de estas. Si en algún momento asignamos algún tipo de dato diferente al que hemos definido al crear la variable se producirá un error.

Para almacenar un valor en una variable solo deberemos escribir una instrucción como esta:

[Nombre Variable] = [Valor]

Valor: será el valor que queremos almacenar en nuestra variable.

Recuerda que si no está definida y no tienes la opción **Option Explicit, Visual Basic** definirá automáticamente la variable como tipo **Variant**.

Podemos decir, para facilitar el entendimiento de la asignación de valores en una variable, que esta se realiza de derecha hacia izquierda.

Tipos de Declaraciones.

Dim: Declara una variable de ámbito local al módulo donde se declara. Si en vez de declararse en un formulario se declarase dentro de un procedimiento o función, solo sería visible a éste, liberándose al salir del procedimiento o módulo donde se declare.
Static: Aún cuando su ámbito es local, declaradas dentro de un módulo, procedimiento o función, no se liberan al salir del ámbito de su declaración, manteniendo el último valor acumulado al volver a ejecuctarse el módulo o procedimiento que lo creó.
Public: Son variables públicas y son accesibles desde todos los formularios de la aplicación. Es aconsejable declararlas en el formulario o módulo de código Inicial, para que sean visibles desde todos los formularios. No puede declarar variables públicas en un procedimiento, sólo en la sección Declaraciones de un módulo

Tipo	Tamaño	Rango
Boolean	Números de 16 bits (2 bytes)	Pueden almacenar los valores True o False.
Byte	Números de 8 bits (1 byte)	Sin signo con un intervalo de valores entre 0 y 225.
Integer	Números de 16 bits (2 bytes)	Valores que van de -32.768 a 32.767.
Long	Números con signo de 32 bits (4 bytes)	Valores comprendidos entre -2.147.483.648 y 2.147.483.647
Single	Números IEEE de coma flotante de 32 bits (4 bytes)	Valores que van de -3,402823E38 a -1,401298E-45 para valores negativos y de 1,401298E-45 a 3,402823E38 para valores positivos.
Double	Se almacenan como números IEEE de coma flotante de 64 bits (8 bytes)	Valores de -1,79769313486232E308 a -4,94065645841247E-324 para valores negativos y de 4,94065645841247E-324 a 1,79769313486232E308 para valores positivos.
Currency	Números de 64 bits (8 bytes)	Formato de número entero de punto fijo con 15 dígitos a la izquierda del signo decimal y 4 dígitos a la derecha. Proporciona un intervalo de -922.337.203.685.477,5808 a 922.337.203.685.477,5807.
String	De longitud variable	Pueden contener hasta 2.000 millones de caracteres (2^31).

	De longitud fija	Pueden contener de 1 a 64 KB (2^16) caracteres.
Date	Números IEEE de signo flotante de 64 bits (8 bytes)	Del 1 de enero del 100 al 31 de diciembre de 9999 y horarios de 0:00:00 a 23:59:59. Cualquier valor reconocible de fecha literal se puede asignar a las variables tipo Date. Los literales de fecha se deben poner entre caracteres de signo de número (#).

Operadores:

Los Operadores son instrucciones usadas para cálculos matemáticos o lógicos. Los operadores se pueden dividir en cinco grupos, aritméticos, concatenación, asignación, comparación y lógicos

Los operadores se situan siempre entre los dos valores con los que realiza su cometido, en la siguiente tabla se pueden conocer sus funciones.

Operador Operación que realiza

Aritméticos

+	Realiza las funciones de suma entre dos valores. ResultadoSuma = valor_1 + valor_2
-	Resta dos valores. ResultadoResta = valor_1 - valor_2
*****	Multiplica dos valores. ResultadoMultiplicación = valor_1 * valor_2
****	División entera entre dos números. ResultadoDivEntera = valor_Dividendo \ valor_Divisor
/	División en coma flotante entre dos números. ResDivComaFlotante = valor_Dividendo / valor_Divisor
Mod	Devuelve el resto tras una operación de división. RestoDivision = valor_Dividendo Mod valor_Divisor
^	Se utiliza para elevar un número a la potencia del exponente. ResultadoExponente = valor^exponente

Concatenación

&	Concatena dos valores. Si los valores a usar son numéricos los convierte a cadena de caracteres. ResultadoString = valorString_1 & valorLong_2
+	Realiza la concatenación de dos valores. Cuando utilice el operador + , quizá no pueda determinar si se va a realizar una suma o una concatenación de cadenas. Utilice el operador & para la concatenación, de modo que se eviten ambigüedades y se suministren programas claros y explícitos. Resultado_1_n = valor_1 + valor_n

Asignación

=	Asigna a la parte izquierda de la expresión el valor de la parte derecha. Valor_1 = valor_n

Comparación

>	Mayor que... Verdadero si la parte Izquierda de la expresión es mayor que la parte derecha. Verdadero -> 7 > 3 Falso -> 3 > 7
<	Menor que... Verdadero si la parte Izquierda de la expresión es menor que la parte derecha. Verdadero -> 3 < 7 Falso -> 7 < 3
=	Igual que... Verdadero si la parte Izquierda de la expresión es igual que la parte derecha.

	Verdadero -> 2 = 2 Falso -> 2 = 3
> =	Mayor o igual que... Verdadero si la parte Izquierda de la expresión es mayor o igual que la parte derecha. Verdadero -> 2 >= 2 Verdadero -> 2 >= 1 Falso -> 2 >= 3
< =	Menor o igual que... Verdadero si la parte Izquierda de la expresión es menor o igual que la parte derecha. Verdadero -> 2 <= 2 Verdadero -> 2 <= 3 Falso -> 2 <= 1
< >	Distinto de... Verdadero si ambas partes de la expresión son distintas. Verdadero -> 2 <> 3 Falso -> 2 <> 2
Is	Se utiliza para comparar dos variables de referencia de objeto. Verdadero si ambas expresiones hacen referencia al mismo objeto. Resultado = objeto1 Is objeto2

Lógicos

And Verdadero si ambos valores de la expresión son verdaderos.
Verdadero And Verdadero --> Verdadero
Verdadero And Falso --> Falso
Falso And Verdadero --> Falso
Falso And Falso --> Falso

Or Verdadero si alguno o ambos valores de la expresión son verdadero.
Verdadero Or Verdadero --> Verdadero
Verdadero Or Falso --> Verdadero
Falso Or Verdadero --> Verdadero
Falso Or Falso --> Falso

Not Devuelve el valor contrario al de la expresión.
MiValor = Not Falso --> MiValor = Verdadero
MiValor = Not Verdadero --> MiValor = Falso

Xor Realiza una exclusión lógica entre dos expresiones.
Es veradero si solo una de las partes de la expresión es verdadero.
Verdadero Xor Verdadero --> Falso
Verdadero Xor Falso --> Verdadero
Falso Xor Verdadero --> Verdadero
Falso Xor Falso --> Falso

Eqv Se utiliza para efectuar una equivalencia lógica de dos expresiones.
Es Falso si solo una de las partes de la expresión es Falso.
Verdadero Eqv Verdadero --> Verdadero
Verdadero Eqv Falso --> Falso
Falso Eqv Verdadero --> Falso
Falso Eqv Falso --> Verdadero

FUNCIONES DE CADENAS

Se denomina CADENA a una sucesión de caracteres. Una cadena puede tener uno o varios caracteres alfanuméricos. Una cadena es también una sucesión de números.

Ejemplo de cadenas:

Curso de Visual Basic
abcdefghijklmnopqrstuvwxyz1234567890
123456789

Funciones con cadenas:

Str (número**)** Convierte un número a una cadena en numeración decimal.
Val (cadena numérica**)** Obtiene el valor (el número) correspondiente a esa cadena.

Ejemplos:
Variablenumerica = **Val** (TextBox1.Text)
Este ejemplo convierte la cadena de caracteres (numéricos) que hubiese en la caja de texto TextBox1 en un número, y asocia este número a la variable *Variablenumerica*.
Si el contenido de la caja de textos no fuesen caracteres numérico (abcd, por ejemplo), *Variablenumerica* tomaría el valor 0.
Label1.Caption = **Str** (*Variablenumerica*)
Este ejemplo pondría en la etiqueta Label1 los caracteres correspondientes al valor que tuviese la variable *Variablenumerica* .

Mas funciones de cadena:
Left (cadena, n) Extrae los n primeros caracteres de una cadena, comenzando por la izquierda.
Si cadena = Curso de Visual Basic (Para todos los ejemplos)
Resultado = **Left** (cadena, 10) ---->Resultado = Curso de V
Right (cadena, n) Extrae lo n últimos caracteres de la cadena
Resultado = **Right** (cadena, 10) ---->Resultado = sual Basic
Mid (cadena, m, n) Extrae n caracteres de la cadena, siendo el primer carácter extraído el que ocupa el lugar m.
Resultado = **Mid** (cadena, 3, 10) ---->Resultado = rso de Vis
LCase (cadena) Devuelve otra cadena igual, pero con todos los caracteres en minúsculas. (LCase = Lower Case)

Resultado = **Lcase** (cadena) ---->Resultado = curso de visual basic
UCase (cadena) Devuelve otra cadena igual, pero con todos los caracteres en mayúsculas. (UCase = Upper Case)
Resultado = **UCase** (cadena) ---->Resultado = CURSO DE VISUAL BASIC
Len (cadena) Devuelve la longitud de la cadena
Resultado = **Len** (cadena) ---->Resultado = 21
LenB (Cadena) Devuelve el número de Bytes empleados para almacenar la cadena. Sorpréndase, es el doble que Len (Cadena)

String (n, carácter) Devuelve una cadena de n caracteres como el indicado
Resultado = **String** (8, "a") ---->Resultado = aaaaaaaa
Resultado = **String** (8, Chr(65)) ---->Resultado = AAAAAAAA
Resultado = **String** (8, 65) ---->Resultado = AAAAAAAA
Space (n) Devuelve una cadena formada por n espacios.
Resultado = "A" + **Space** (6)+ "B"---->Resultado = A B
LTrim Elimina los posibles espacios que tenga una cadena por su izquierda.
Rtrim Elimina los posibles espacios que tenga una cadena por su derecha.
Trim Elimina los espacios que tenga una cadena, tanto por su izquierda como por su derecha. (No elimina los espacios centrales de la cadena)
Estas tres funciones se emplean para quitar los posibles espacios que pueden resultar de una entrada de datos. Tienen especial importancia cuando se toman los datos de un archivo o base de datos, donde fueron introducidos por otro programa.
Función FORMAT
Esta función permite presentar cadenas de numéricas o fechas de una determinada forma. Permite establecer el **Format**o de esa cadena.

Format(expresión[, formato[, primerdíadesemana[, primerasemanadelaño]]])

Para poner los números separados por millares :
Variable = **Format**(1234567, "##,###,###") Variable = 1.234.567
(Cada carácter # indica que ahí va un número. El separador debe ser una coma, no un punto, aunque esto depende del idioma que esté usando)

LA CAJA DE MENSAJES. MESSAGEBOX O MSGBOX

Las cajas de mensajes o MessageBox, tienen una función clara, que es la de mostrar una determinada información, aviso, o pregunta para que el usuario tenga conocimiento de ella y actúe.

Hay 2 formas diferentes de mostrar información:

1 - El aviso es sí, que tiene por objetivo mostrar una información de interés.

2 - El aviso con espera de respuesta, que muestra una información esperando que el usuario seleccione una de las respuestas posibles para que el programa la trate.

Una caja de mensaje, puede ser por ejemplo, la instrucción ***MsgBox "Hola"***. Por defecto, la caja de mensaje será similar a esta:

Debe darse cuenta de algunas cosas:
En primer lugar el mensaje, *"Hola"* que se escribe a continuación de la palabra *MsgBox*, también debe darse cuenta del botón *Aceptar* que tiene el *Focus* de la ventana activa y que sólo hay ese botón, y por último el título de la ventana.

Podemos modificar estos parámetros para alcanzar nuestros objetivos, por eso, vamos a escribir ahora este código: *MsgBox "Hola", ,"Ejemplo"* .

El resultado es:

Como podemos apreciar en el código, la caja de mensaje posee un título *Ejemplo* y el mensaje, pero es posible que deseemos escribir un mensaje en varias líneas con salto de párrafo. Nada tan fácil como este código por ejemplo: *MsgBox "Hola" & vbCrLf & "Esto es un ejemplo.", , "Ejemplo"*.
El resultado es:

Habrá observado en la expresión anterior que se ha utilizado vbCrLf (Visual Basic Carriage Return Line Feed, VB retorno de carro y avance de línea) Vea mas adelante la aclaración de esta expresión. Con ella logramos introducir un salto de línea.

Supongo que se habrá percatado de que entre el mensaje y el título de la ventana, hemos escrito dos comas, esto es porque entre las comas, debe ir un número que representará el icono a mostrar. Existen cuatro iconos diferentes además de la posibilidad de no mostrar ninguno. Los iconos son:

Estos iconos corresponden a los siguientes mensajes:
Mensaje crítico.
Mensaje de pregunta.
Mensaje exclamativo.
Mensaje de información.

(Sólo en W32. En Windows 3.xx dispone de otros diferentes, aunque con el mismo significado)

Para mostrar el icono en cuestión o para que Visual Basic lo entienda, es necesario escribir lo siguiente:

Mensaje crítico.	VbCritical	ó	16
Mensaje de pregunta.	VbQuestion	ó	32
Mensaje exclamativo.	VbExcalamqtion	ó	48
Mensaje de información.	VbInformation	ó	64

Note que es lo mismo insertar VbCritical o 16.

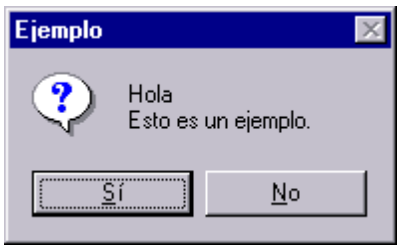
Vamos a ver un ejemplo añadiendo un icono al último ejemplo:
MsgBox "Hola" & vbCrLf & "Esto es un ejemplo.", VbQuestion , "Ejemplo"

El resultado es:

Ahora bien, es posible que queramos mostrar algún otro botón que o bien no sea el de *Aceptar* o que además del botón de *Aceptar* haya más botones. Para este propósito, tenemos los siguientes parámetros:

Aceptar	vbOKOnly	ó	0
Aceptar y Cancelar	vbOKCancel	ó	1
Anular, Reintentar, Ignorar	vbAbortRetryIgnore	ó	2
Sí, No y Cancelar	vbYesNoCancel	ó	3
Sí y No	vbYesNo	ó	4
Reintentar y Cancelar	vbRetryCancel	ó	5
Aplicación modal	vbApplicationModal	ó	0 (Es la caja de mensaje sin icono)

La forma de hacer esto es sumar al parámetro del icono que queremos mostrar el valor de los botones que deseamos que aparezcan.
Así por ejemplo:
MsgBox "Hola" & vbCrLf & "Esto es un ejemplo.", VbQuestion + vbYesNo , "Ejemplo"
El resultado es:

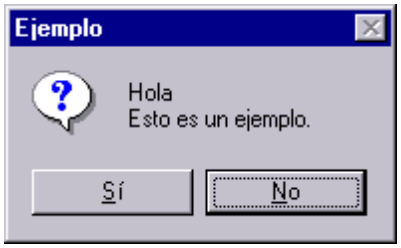


Aún así, es posible que deseemos que el *Focus* lo adquiera otro un botón determinado. Por ejemplo, en este caso el *Focus* lo tiene el botón *Sí*, pero es posible que deseemos que lo tenga el botón *No* por ejemplo. Esto se consigue con los siguientes parámetros:

Primer botón predeterminado	vbDefaultButton1	ó	0
Segundo botón predeterminado	vbDefaultButton2	ó	256
Tercer botón predeterminado	vbDefaultButton3	ó	512

Por ejemplo: MsgBox "Hola" & vbCrLf & "Esto es un ejemplo.", VbQuestion + vbYesNo + vbDefaultButton2, "Ejemplo"

El resultado es:



Si no se señala el botón predeterminado, Visual Basic seleccionará el primer botón. En caso de seleccionar como predeterminado un botón que no existe, (por ejemplo el tercero), Visual Basic seleccionará el primero.

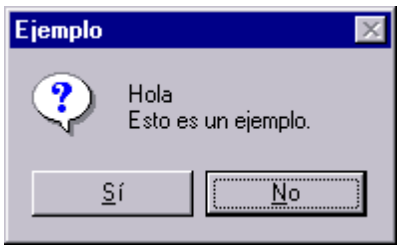
Ahora bien, si decidimos mostrar un mensaje esperando una respuesta, o queremos saber que botón ha pulsado el usuario, esto lo podemos conseguir mediante el siguiente código de respuestas:

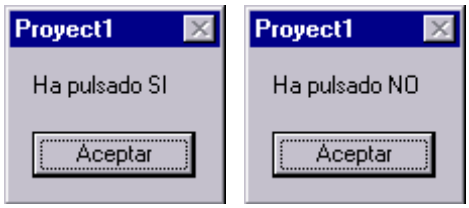
Aceptar	vbOK	ó	1
Cancelar	vbCancel	ó	2
Anular	vbAbort	ó	3
Reintentar	vbRetry	ó	4
Ignorar	vbIgnore	ó	5
Sí	vbYes	ó	6
No	vbNo	ó	7

Así por ejemplo, el siguiente código:

```
Dim Resp As Integer

Resp = MsgBox("Hola" & vbCrLf & "Esto es un ejemplo.", VbQuestion + vbYesNo +
vbDefaultButton2, "Ejemplo")

If Resp = 6 Then
   MsgBox "Ha pulsado SI"
Else
   MsgBox "Ha pulsado NO"
End If
```

Tiene el resultado siguiente:





Si pulsamos el botón *Sí* obtendremos una acción, y si pulsamos el otro botón otra acción.

Ahora bien, para elegir o seleccionar un evento o acción, el usuario debe saber combinar los códigos, sabiendo que un *MsgBox* posee la siguiente sintaxis principal:

MsgBox Mensaje, Botones, Título de la ventana

InputBox

El InputBox o caja de entrada es otra de las partes más utilizadas para la interacción del usuario con la aplicación. Es importante que el usuario interactúe con la aplicación para ser el protagonista de esta.

El InputBox nos permite sacar una caja donde el usuario pasará un parámetro, valor o dato para que el programa lo trate y lo ejecute.

El mensaje que quiere que aparezca se realiza de forma casi idéntica al MessageBox. Puede escribirse varias líneas de texto seguidas por la constante de Visual Basic vbCrLf o salto de línea o párrafo.

La sentencia es: *Val* = InputBox (Mensaje, Titulo, ValorPredeterminado)

Val almacenará el texto escrito por el usuario, que puede ser una cantidad, cadena string, ... etc.

Por ejemplo:
Dim Val As String

Val = InputBox("Deme su nombre", "Ejemplo")
MsgBox "Su nombre es: " & Val

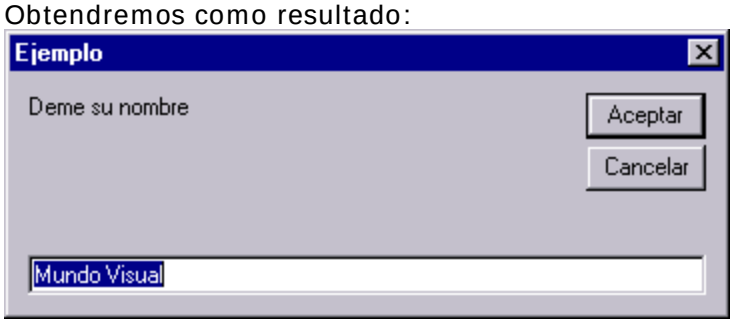
Tiene como resultado:

(El usuario teclea el nombre - Mundo Visual - y hace click en Aceptar. A continuación se muestra el MsgBox - 2ª línea del código anterior)

Ahora bien, podemos determinar un texto predeterminado a la caja de entrada, como por ejemplo:

Dim Val As String

Val = InputBox("Deme su nombre", "Ejemplo", "Mundo Visual")
MsgBox "Su nombre es: " & Val
Obtendremos como resultado:



Es importante determinar que si el usuario elige el botón *Cancelar*, el programa devolverá una cadena de caracteres igual a 0, es decir, *Val =""*.

La caja de entrada puede ser sin embargo más personalizada mediante dos parámetros como son la posición de la ventana de entrada de datos en la pantalla. Estos parámetros se ponen a continuación del *ValorPredeterminado*.
Por ejemplo:

Dim Val As String
Val = InputBox("Deme su nombre", "Ejemplo", "Mundo Visual", 1200, 1400)
Situará la ventana en el eje de las *X* a 1200 Twips (posición horizontal) y la *Y* a 1400 twips (posición vertical).

LENGUAJE BASIC DEL VISUAL BASIC.

Estructuras de decisión

Las líneas de un programa, dentro de un evento, se ejecutan de arriba abajo. Pero en muchas ocasiones no nos interesa esta linealidad con lo que podemos cambiar el orden de las líneas de código según el entorno. Estas estructuras nos permiten tomar decisiones según las condiciones que se den en nuestra aplicación. Por ejemplo, podemos tener una aplicación en la que nos interese calcular el peso ideal de una persona dada, su estatura y su sexo. Pues bien, el sexo será la condición que marcará que camino hay que tomar, ya que si es un hombre tendremos que multiplicar su estatura por una valor y si es una mujer por otra.

Vamos a ir viendo las principales estructuras de decisión una a una.

If...Then...End If

La estructura básica de esta instrucción es la siguiente:

```
If Condición Then
[Instrucciones Verdadero]
Else
[Instrucciones Falso]
End If
```

Los corchetes muestran partes opcionales de la instrucción.

Condición: Aquí escribiremos la condición a evaluar, para que la computadora nos devuelva una respuesta y según esta respuesta podamos actuar. Por ejemplo: sitúate en el ejemplo anterior. Nosotros podemos tener un **TextBox** llamado sexo, en el que escribiremos una "V" para varón o una "H" para hembra. En este lugar reservado para la condición podremos escribir **Sexo.Text = "H"**, quedando la línea completa con el **If** incluido de la siguiente manera **If Sexo.Text = "H" Then**. Fíjate que aquí no estamos haciendo una asignación, si no que lo que estamos haciendo es plantearnos una pregunta que podemos leer más o menos de la siguiente forma: **Si el contenido de Sexo es H, entonces...** a lo que la computadora nos devolverá verdadero o falso. Según esta respuesta nosotros actuaremos de una forma u otra.

[Instrucciones Verdadero]: Aquí pondremos la instrucción o instrucciones que se deberá realizar si la respuesta a la condición es Verdadera.

Si nosotros solo tenemos una instrucción para colocar en el lugar de **[Instrucciones Verdadero]** y ninguna en **[Instrucciones Falso]** podemos escribir esta estructura de la siguiente manera:

If [Condición] Then [Instrucción Verdadero]

Recuerda que en esta estructura solo se puede escribir una instrucción verdadera.

Si nosotros lo que queremos escribir son varias instrucciones, pero no queremos que la computadora realice nada en el caso de que la condición sea Falsa podemos escribir la estructura de la siguiente forma:

```
If [Condición] Then
[Instrucciones Verdadero]
End If
```

SELECT CASE

Esta estructura la utilizaremos en los casos en los que tengamos muchas condiciones a evaluar, ya que con la estructura **If** se podría complicar bastante.

En esta nueva estructura de decisión se valoran los diferentes valores que puede tomar una determinada expresión y según el valor que tenga esta se actúa en consecuencia.

```
Select Case [Expresión para comparar]
Case [Expresión 1]
        [Instrucciones 1]
...
Case [Expresión n]
        [Instrucciones n]
[Case Else]
        [Instrucciones Else]
End Select
```

En **[Expresión para comparar]** pondremos la expresión sobre la cual queremos preguntar.

En **[Expresión 1]** escribiremos cual es la pregunta que deseamos hacer sobre el valor escrito anteriormente.

En **[Instrucciones 1]** pondremos las instrucciones que se realizarán en caso de que las **[Expresión 1]** sea verdadera.

Podremos poner tantas **[Expresiones]** como queramos.

Si queremos que se haga algo en caso que ninguna de las expresiones que hemos puesto anteriormente se cumpla, podemos escribir **[Case Else]** y seguidamente la o las instrucciones que se tienen que ejecutar en dicho caso.

No te olvides, igual que en el caso del **If**, cerrar la expresión utilizando **End Select**.

INTRODUCCIÓN A LAS ESTRUCTURAS DE REPETICIÓN

El número de veces que se repetirá la instrucción o instrucciones puede depender de un contador o de una condición.

En esta lección vamos a ver los dos tipos de bucles: con contador o con condiciones.

FOR... NEXT

Esta es una estructura de repetición o bucle, la cual depende de un contador que nos controla el número de veces que se deberá repetir una instrucción o varias instrucciones.

En esta estructura siempre deberemos especificar la variable (contador), un valor inicial y un valor final. Normalmente el contador incrementará de uno en uno a no ser que nosotros indiquemos lo contrario.

La estructura del bucle utilizando contador es la siguiente:

For ***Contador*** **=** ***Inicio*** **To** ***Fin*** **[Step** ***Incremento*****]**
    [Instrucciones]
Next ***Contador***

Vamos a explicar las diferentes partes de esta estructura:

Contador: Aquí es donde nosotros escribiremos el nombre de la variable que queremos utilizar como contador.

Inicio: Valor inicial de la variable.

Fin: Valor final de la variable. Cuando la variable llegue a este valor, el bucle no se volverá a ejecutar.

Step: Esta instrucción es opcional. Si no la ponemos el contador irá incrementando de uno en uno. Si especificamos un número detrás de **Step** hacemos que nuestro contador aumente un número determinado de pasos.

Incremento: Número que marcará los pasos que debe aumentar el contador. Este número puede ser tanto positivo como negativo. Eso sí, siempre hemos de tener cuidado con los valores iniciales y finales para que no se produzca ningún tipo de error. No podemos hacer, por ejemplo, que el valor inicial sea 10 y el final 1 siempre y cuando no pongamos como **step** un valor negativo.

Instrucciones: Aquí escribiremos la o las instrucciones que queremos que se repitan.

Next Contador: Línea que indica que se termina el bucle y hace que aumente el contador según nos indique **step**.

DO... LOOP

Ahora vamos a ver un tipo de estructura de repetición que depende de una condición. Las instrucciones que hay dentro del bucle se repiten **mientras se cumpla la condición**, mientras la condición sea **Verdadera**.

Tenemos dos tipos de estructuras **Do...Loop**, una en la que se mira la condición antes de realizar ninguna instrucción y otra que se mira después de realizar, al menos, una vez las instrucciones de dentro del bucle.

Vamos a ver las dos estructuras y después pasaremos a comentar sus diferencias:

```
Do While [Condición]        Do
[Instrucciones]                 [Instrucciones]
Loop                        Loop While [Condición]
```

Vamos a comentar las diferentes partes de estas estructuras.

Condición: lugar reservado para colocar la pregunta que queremos realizar para ver si es **verdadero** o **falso**.

Instrucciones: líneas de código que se ejecutan mientras que la condición sea verdadera.

En la primera estructura de repetición lo primero que se mira es la condición, si esta se cumple pasamos a realizar las instrucciones que tenemos en el interior del bucle, si no se cumple nos saltamos todas las instrucciones hasta llegar al **Loop** que nos indica el final de dicho bucle.

La segunda estructura de repetición es diferente, primero entramos en el bucle y realizamos todas las instrucciones una vez, después miramos la condición, si esta se cumple volvemos a realizar las instrucciones que tenemos dentro del bucle, por lo contrario si esta no se cumple salimos del bucle.

Es difícil explicar en que momentos se necesitará una u otra instrucción ya que esto dependerá de cada caso y nada mejor que aprenderlo sobre la marcha.

Existen dos estructuras a las que hemos visto antes pero con la diferencia que el bucle se repetirá **mientras no se cumpla la condición**, mientras la condición sea **Falsa**.

La estructura sería la siguiente:

```
Do Until [Condición]        Do
[Instrucciones]                 [Instrucciones]
Loop                        Loop Until [Condición]
```

Observa la diferencia de estas dos estructuras con las vistas anteriormente.

La única diferencia es que en las primeras utilizamos la palabra, **While** y en estas últimas **Until**, por lo demás todo el "funcionamiento" es exactamente igual.

BUCLES ANIDADOS

Es anidar, poner dentro de otro, diferentes estructuras de repetición.

CREACIÓN DE MENÚS.
DISEÑO DE UN MENÚ

Para diseñar un menú se utilizará Herramientas/Editor de Menús. Los elementos de un menú pueden ser órdenes o comandos, que al hacer clic sobre ellos ejecutan acciones; submenús, que despliegan una nueva lista de elementos y; separadores o línea horizontal que agrupa las opciones del menú. Una vez en la ventana de diseño de menús, se introducirán los siguientes datos:

Caption Nombre del menú que se desea crear (lo que ve el usuario). Se insertará un ampersand (&) delante de la letra que dará acceso al menú. Para introducir un separador, se escribirá un guión (-) en este apartado.

Name Nombre utilizado en el código para referirse al menú.

Index Permite que un conjunto de órdenes sean agrupadas en una matriz de controles.

ShortCut Permite definir un acelerador, es decir, una combinación de teclas para activar el elemento.

Checked Útil para indicar si una orden está activa o no. Si lo está, en la orden aparece una marca __ a su izquierda.

Enabled Útil para desactivar una orden. Si una orden tiene esta propiedad marcada, aparecerá atenuada y no podrá ser ejecutada.

Visible Es útil cuando en tiempo de ejecución se quiere ocultar una orden.

WindowList Permite especificar si un menú mantiene una lista de las ventanas abiertas, con una marca Ö a la izquierda de la ventana activa.

HelpContexID Se utiliza para proveer una ayuda en línea para una aplicación.

NegotiatePosition Determina si el menú va a aparecer en el formulario y cómo lo hace.

Para introducir sangrías entre los elementos de menú, se pulsará los botones de flecha. El botón Next inserta un nuevo elemento de menú o, se desplaza al siguiente.

En tiempo de ejecución se podrán añadir órdenes a un menú. Para ello se utilizará la sentencia Load, tal como se utilizaba en matrices de controles. Para eliminar una orden de un menú, se utilizará Unload. Para poder utilizar estas dos sentencias, las órdenes que componen el menú tienen que pertenecer a una matriz de controles, por lo que durante el diseño se debe haber creado al menos, un elemento (haber puesto a un comando la propiedad Index a 0).

MENÚS DESPLEGABLES FLOTANTES O EMERGENTES

Para poder crear un menú emergente o contextual, se utilizará el método:

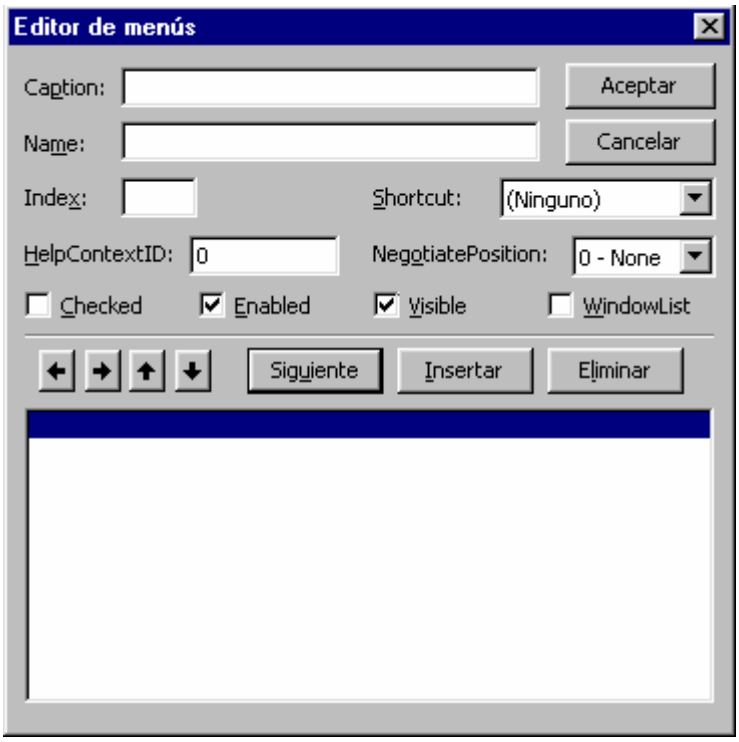
[formulario.] PopupMenu NombreMenú

donde NombreMenú es el valor de la propiedad Name del menú que se quiere mostrar. Ya se verá cómo hacer que aparezcan al pulsar el botón derecho del ratón.

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.

También puedes poner en funcionamiento el **Editor de menús** utilizando la combinación de teclas **[Control] + [E]** o utilizando la **Barra de herramientas estándar** haciendo clic sobre este 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 parte 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. 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** deb

MATRICES

Las matrices son un grupo de valores que tienen un mismo nombre y se diferencian entre ellas por el lugar que ocupan. A este lugar que ocupa se le llama Indice. Gracias a este índice podemos crear un código, utilizando estructuras repetitivas que nos ayuden a trabajar con estos datos, ahorrando de esta manera código.

Normalmente, las matrices se definen con un límite inferior y uno superior. Con la resta de ambos tenemos el número de elementos que pueden entrar dentro de la matriz. Tenemos que pensar que si nosotros solo definimos el valor superior, el primer objeto tendrá como índice 0 y el último el número que nosotros hayamos definido como límite superior. De esta forma, si nosotros definimos una matriz de una sola dimensión con límete superior 5, en realidad tenemos 6 objetos con índices 0, 1, 2, 3, 4, 5.

Al igual que las variables y constantes, en las tablas también se tiene que definir el tipo de valor vamos a almacenar dentro.

Las matrices podemos definirlas de solo una dimensión, como si se tratase de una gran fila de datos y en otras muchas ocasiones nos puede interesar utilizar estructuras de dos dimensiones o incluso más, en las que buscaremos los datos por la fila y la columna que ocupan. En este caso estas matrices tendrán 2 índices.

Por ejemplo en el caso de un tablero de ajedrez nos interesa saber en que fila y en que columna se encuentra situado una ficha determinada para saber si el movimiento que deseamos realizar está o no permitido.

Como definir una matriz

Podemos decir que una matriz se define exactamente igual que una variable y una constante, pero con la diferencia que deberemos especificar el tamaño que deseamos que tenga. Esto lo hacemos de la siguiente forma, según si la matriz es de una o más dimensiones:

Matriz de una dimensión

Dim [Nombre Matriz] ([Tamaño]) As [Tipo de datos]

Tamaño: aquí definiremos, siempre entre paréntesis, el tamaño de nuestra matriz. Debemos recordar que si nosotros definimos una matriz de 5 elementos, el primer elemento está ocupando la posición 0 y el último la 5, por lo tanto en realidad estamos trabajando con 6 elementos.

Si nosotros deseamos empezar a trabajar desde un número determinado de índice hasta otro deberemos definir la matriz de la siguiente manera:

Dim [Nombre Matriz] ([Índice inicial] To [Índice final]) As [Tipo de datos]

Una matriz definida de esta forma tendrá como primer índice el Índice inicial y como último el Índice final.

Matriz de más dimensiones

Dim [Nombre Matriz] ([Tamaño fila], [Tamaño columna]) As [Tipo de datos]

Una matriz de más de una dimensión podemos pensar que es como una cuadricula en la que necesitaremos dos o más índices para hacer referencia a alguno de los objetos que tenemos en su interior.

Tamaño fila: aquí definiremos el número de filas que deseamos tenga nuestra matriz.

Tamaño columna: aquí definiremos el número de columnas que deseamos tenga nuestra matriz.

Vamos ha hacer una representación gráfica de una matriz con **5 filas** y **10 columnas**:

		Columnas									
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Filas	**1**										
	2										
	3										
	4										
	5										

Observa como las **filas** están dispuestas **horizontalmente**, mientras que las **columnas** lo hacen **verticalmente**.

Así de esta forma, en el momento en el que nosotros queremos hacer referencia a uno de los elementos introducidos en la matriz, deberemos indicar el número de **fila** y el de **columna** que ocupa dicho elemento.

Al igual que en las tablas de una sola dimensión podemos especificar donde queremos que empiece el índice de nuestra matriz, pero en este caso tendremos que especificar cada uno de los índices que utilizamos (fila y columna).

Esta podría ser la definición de una matriz de dos dimensiones definiendo tanto el inicio como el final del índice.

Dim [Nombre Matriz] ([Índice inicial fila] To [Índice final fila], [Índice inicial columna] To [Índice final columna]) As [Tipo de datos]

Matrices

Para asignar un valor a una posición de una matriz, lo haremos exactamente igual que en la asignación de valores en una variables, pero con la diferencia que aquí tenemos que especificar el índice (posición) donde deseamos se almacene el valor.

En el caso de una matriz de una sola dimensión lo deberemos hacer de esta forma:
[Nombre Matriz] ([Indice]) = [Valor]

En caso de una matriz de dos dimensiones lo deberemos hacer de esta otra forma:
[Nombre Matriz] ([Indice fila], [Indice columna]) = [Valor]

En el caso de las matrices deberás tener mucho cuidado en no asignar valores a índices que están fuera de la tabla.

Más adelante, cuando veamos las estructuras de repetición, veremos algunas formas de poder movernos por toda la matriz de forma sencilla para poder realizar cálculos, ordenaciones de datos y búsquedas de datos en una matriz.

Matrices de controles

En este capitulo vamos ha ver un sistema con el cual nos podemos ahorrar algunas líneas de código. Sobre todo en el momento en que tenemos muchos objetos que son exactamente iguales y queremos que todos actúen de una misma forma.

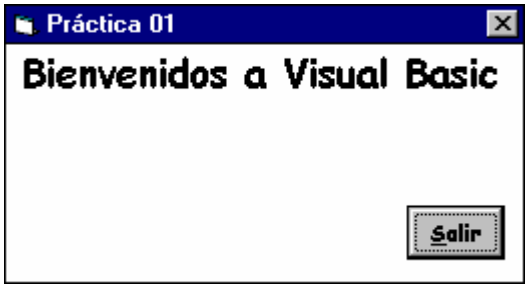
BIBLIOGRAFÍA:

Julio Vásquez Paragulla
Diseño de Programación
Editorial San Marcos
1997-Lima
Joel Carrasco Muñoz, César Bustamante Gtz
Desarrollo de Aplicaciones en Visual Basic
Universidad nacional de Ingeniería
1997-Lima
Bruce McKinney
Programación Avanzada M. Visual Basic
Colección Mc Graw Hill
1998-España
Websites:
http://www.programación.net
http://www.canalvisualbasic.net/

Recuperó la información: Lic. Salazar Córdova Héctor

EJERCICIO Nº 01

1. Elaborar un Programa que salude al usuario:



Solución

-Cambio de las propiedades de los controles:

Control	Propiedad	Valor
Form	Autoredraw	True
	Caption	Práctica 01
Command1	Name	CmdSalir
	Caption	&Salir

```
Private Sub CmdSalir_Click()
 End
End Sub
```

EJERCICIO Nº 02

CALCULAR EL ÁREA DE UN TRIÁNGULO CONOCIENDO SUS TRES LADOS, APLICANDO LA SIGUIENTE FORMULA:

$$Area = \sqrt{p(p-a)(p-b)(p-c)}$$

P=(A+B+C)/2

DONDE: P= SEMIPERÍMETRO, A,B,C: LADOS

SOLUCIÓN:

CONTROL	PROPIEDAD	VALOR
LABEL1	CAPTION	VALOR A
LABEL2	CAPTION	VALOR B
LABEL3	CAPTION	VALOR C
LABEL4	CAPTION	AREA
TEXT1	TEXT	NADA
	NAME	TXTA
TEXT2	TEXT	NADA
	NAME	TXTB
TEXT3	TEXT	NADA
	NAME	TXTC
TEXT4	TEXT	NADA
	NAME	TXTAREA

```
PRIVATE SUB TXTA_KEYPRESS(KEYASCII AS INTEGER)
 IF KEYASCII=13 THEN
   TXTB.SETFOCUS
END IF
END SUB

PRIVATE SUB CMDEJECUTAR_CLICK()
DIM A AS INTEGER, B AS INTEGER, C AS INTEGER
DIM P AS SINGLE, AREA AS SINGLE
A = VAL(TXTA.TEXT)
B = VAL(TXTB.TEXT)
C = VAL(TXTC.TEXT)
P = (A + B + C) / 2    'CALCULO DEL SEMIPERÍMETRO
AREA = SQR(P * (P - A) * (P - B) * (P - C))
```

```
TXTAREA.TEXT = STR(AREA)
END SUB
```

EJERCICIO Nº 03

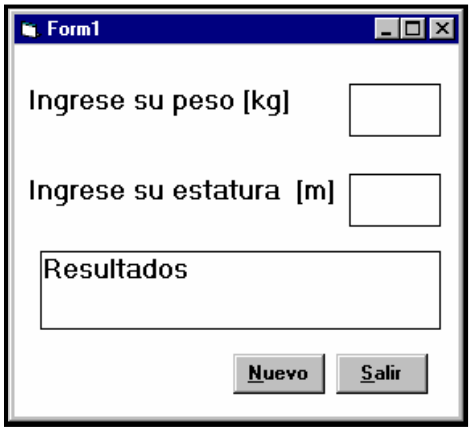
DISEÑAR Y ELABORAR UN PROGRAMA QUE PERMITA DECIDIR, SI UN USUARIO ESTA POSIBLEMENTE DESNUTRIDO, NORMAL U OBESO DE ACUERDO A LA SIGUIENTE VARIABLE $K=P/M^2$. SEGÚN EL RANGO:

K<19 POSIBLEMENTE DESNUTRIDO
K [19; 24] NORMAL
K>24 POSIBLEMENTE OBESO

SOLUCIÓN:

CONTROL	PROPIEDAD	VALOR
LABEL1	CAPTION	INGRESE SU PESO [KG]
LABEL2	CAPTION	INGRESE SU ESTATURA[M]
LABEL3	CAPTION	RESULTADOS
	NAME	LBLRESUL
TEXT1	TEXT	NADA
	NAME	TXTP
TEXT2	TEXT	NADA
	NAME	TXTE

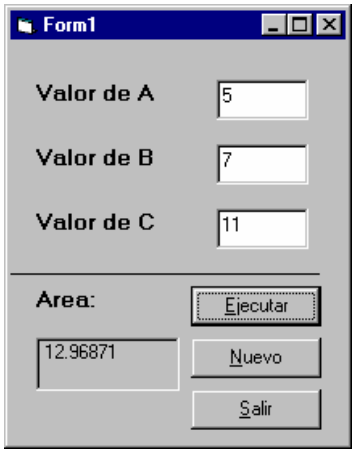
```
OPTION EXPLICIT
 DIM E AS SINGLE, P AS SINGLE, K AS SINGLE

PRIVATE SUB TXTP_KEYPRESS(KEYASCII AS INTEGER)
IF KEYASCII = 13 THEN
 P = VAL(TXTP.TEXT)
 IF P > 0 AND P <= 200 THEN
  TXTE.SETFOCUS
 ELSE
  MSGBOX "INGRESE NUEVAMENTE", VBCRITICAL
 END IF
END IF
END SUB

PRIVATE SUB TXTE_KEYPRESS(KEYASCII AS INTEGER)
IF KEYASCII = 13 THEN
 E = VAL(TXTE.TEXT)
 IF E > 0 AND E <= 2.5 THEN
  K = P / (E ^ 2)
  IF K < 19 THEN
    LBLRESUL.CAPTION = "POSIBLE DESNUTRICIÓN"
  END IF
  IF K >= 19 AND K <= 24 THEN
   LBLRESUL.CAPTION = "ESTA DENTRO DEL RANGO DE NORMAL"
  END IF
  IF K > 24 THEN
    LBLRESUL.CAPTION = "POSIBLE OBESIDAD"
  END IF
 ELSE
   MSGBOX "INGRESE NUEVAMENTE", VBCRITICAL
 END IF
END IF
END SUB
```



EJERCICIO Nº 04

Diseñar y codificar un programa que permita calcular el área de un triángulo en función del semiperímetro.

```
Public a As Single, b As Single, c As Single
```




```
Private Sub TxtA_KeyPress(KeyAscii As Integer)
 a = Val(TxtA.Text)
 If KeyAscii = 13 Then
   If a > 0 Then
    TxtB.SetFocus
   End If
 End If
End Sub

Private Sub TxtB_KeyPress(KeyAscii As Integer)
b = Val(TxtB.Text)
If KeyAscii = 13 Then
   If b > 0 Then
     TxtC.SetFocus
   End If
 End If
End Sub

Private Sub TxtC_KeyPress(KeyAscii As Integer)
 c = Val(TxtC.Text)
 If KeyAscii = 13 Then
   If c > 0 Then
    CmdEjecutar.SetFocus
   End If
 End If
End Sub



Private Sub CmdEjecutar_Click()
 Dim P As Single, Area As Single
 If (a + b) > c And (a - b) < c Then
  P = (a + b + c) / 2
  Area = Sqr(P * Abs(P - a) * Abs(P - b) * Abs(P - c))
  LblArea.Caption = Str(Area)
  Else
   LblArea.Caption = "Error"
   CmdNuevo.SetFocus
 End If
End Sub

Private Sub CmdNuevo_Click()
 TxtA.Text = ""
 TxtB.Text = ""
```

```
 TxtC.Text = ""
 LblArea.Caption = ""
 TxtA.SetFocus
End Sub

Private Sub CmdSalir_Click()
 Unload Me
 End
End Sub
```

EJERCICIO Nº 05

Escribir u programa que permita ingresar por teclado 3 valores numéricos e imprimir el número mayor.

```
Private Sub CmdEjecutar_Click()
 Dim A As Integer, B As Integer, C As Integer
 A = Val(TxtA.Text)
 B = Val(TxtB.Text)
 C = Val(TxtC.Text)
 If A < B Then
  If B < C Then
   LblMsj.Caption = C
  Else
   LblMsj.Caption = B
  End If
 Else
 If A < C Then
   LblMsj.Caption = C
  Else
   LblMsj.Caption = A
  End If
 End If
End Sub
```

EJERCICIO Nº 06

$$Area = \sqrt{p(p-a)(p-b)(p-c)}$$

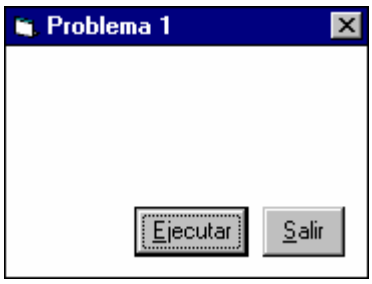
Calcular el área de un triángulo conociendo sus tres lados, aplicando la siguiente formula:

P=(a+b+c)/2
Donde: p= semiperímetro, A,b,c: Lados

Solución:

- Creación de la interfaz
- Colocar los controles
- Cambiar las propiedades de los controles
- Codificar los controles necesarios.

Control	Propiedad	Valor
Form1	Caption	Problema 1
Command1	Caption	&Ejecutar
	Name	CmdEjecutar
Command2	Caption	&Salir
	Name	CmdSalir



Private Sub CmdEjecutar_Click()
Dim a As Integer, b As Integer, c As Integer
Dim P As Single, Area As Single
Print "Area del triángulo"
a = 3
Print "A=" & a
b = 4
Print "B=" & b
c = 5
Print "C=" & c
P = (a + b + c) / 2 'Calculo del Semiperímetro
Area = Sqr(P * (P - a) * (P - b) * (P - c))
Print "Area=" & Area
End Sub

Private Sub CmdSalir_Click()
End
End Sub
Solución 02

Control	Propiedad	Valor
Label1	Caption	Valor A
	Name	LblA
Label2	Caption	Valor B
	Name	LblB
Label3	Caption	Valor C
	Name	LblC
Label4	Caption	Area
	Name	LblArea
Text1	Text	Nada
	Name	TxtA
Text2	Text	Nada
	Name	TxtB
Text3	Text	Nada
	Name	TxtC
Text4	Text	Nada
	Name	TxtArea

Private Sub CmdEjecutar_Click()
Dim a As Integer, b As Integer, c As Integer

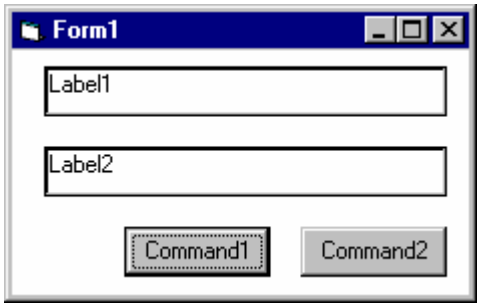
```
Dim P As Single, Area As Single
a = Val(TxtA.Text)
b = Val(TxtB.Text)
c = Val(TxtC.Text)
P = (a + b + c) / 2     'Calculo del Semiperímetro
Area = Sqr(P * (P - a) * (P - b) * (P - c))
TxtArea.Text = Str(Area)
End Sub
```

Problema 02.

Detarminar las raices de una ecuación de Segundo Grado del tipo:
$ax^2+bx+c=0$. Aplicando la fórmula general.

EJERCICIO Nº 08

1.- Elaborar un Programa que calcule la edad actual del usuario, si ingresa por teclado el año que nació.



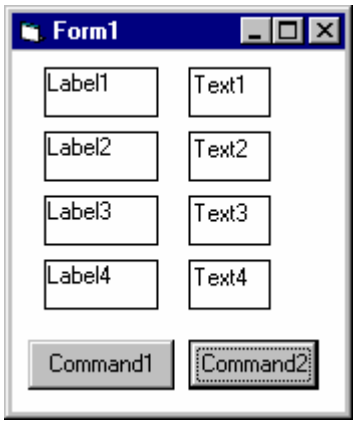
Solución 01

-Cambio de las propiedades de los controles:

Control	Propiedad	Valor
Form	Autoredraw	True
	Caption	Práctica 03
Command1	Name	CmdEjecutar
	Caption	&Ejecutar
Command2	Name	CmdSalir
	Caption	&Salir
Label1	Name	Label1
Label2	Name	Label2

```
Private Sub CmdEjecutar_Click()
Dim Nom As String, Ape As String
Dim Añonacimiento As Integer
Dim Edad As Integer, Añoactual As Integer, a As Integer
Dim Fechactual As Date, Fnac As Date
Nom = InputBox("Tu nombre:")
Ape = InputBox("Tu Apellido:")
Añonacimiento = InputBox("Año que nació:")
Fechactual = Date
Añoactual = Year(Fechactual)
Edad = Añoactual - Añonacimiento
Label1.Caption = "Don/ña. " & Nom & Space(4) & "Ud. tiene " & Edad & "Años"
Label2.Caption = "Sr(a). " & Ape & Space(4) & "hoy es " & Fechactual
End Sub
```

2.- Crear un interfaz que permita calcular el promedio de notas de un estudiante.



Control	Propiedad	Valor
Label1	Name	LblE1
	Caption	Examen1
Label2	Name	LblE2
	Caption	Examen2
Label3	Name	LblPP
	Caption	Practicas
Label4	Name	Lblpro
	Caption	Promedio
Text1	Name	TxtE1
	Text	Nada
Text2	Name	TxtE2
	Text	Nada
Text3	Name	TxtPP
	Text	Nada
Text4	Name	TxtPro
	Text	Nada
Command1	Name	CmdNuevo
	Caption	&Nuevo
Command2	Name	CmdSalir
	Caption	&Salir

Private Sub TxtPP_KeyPress(KeyAscii As Integer)

```
Dim Pro As Integer
 If KeyAscii = 13 Then
  Pro = (Val(TxtE1.Text) + Val(TxtE2.Text) + Val(TxtPP.Text)) / 3
  TxtPro.Text = Str(Pro)
  CmdNuevo.SetFocus
 End If
End Sub
```

Private Sub CmdNuevo_Click()

```
TxtE1.Text = ""
TxtE2.Text = ""
TxtPP.Text = ""
TxtPro.Text = ""
TxtE1.SetFocus
End Sub
```

EJERCICIO Nº 09

1.- Determinar las raíces de una ecuación de segundo grado de tipo $AX^2+BX+C=0$. Considerar la siguiente fórmula:

$$X_{1,2} = \frac{-B \pm \sqrt{B^2 - 4AC}}{2A}$$

Solución:

-Cambio de las propiedades de los controles:

Control	Propiedad	Valor
Form	Autoredraw	True
	Caption	Práctica 04
Command1	Name	CmdEjecutar
	Caption	&Ejecutar
Command2	Name	CmdSalir
	Caption	&Salir

Codificación:

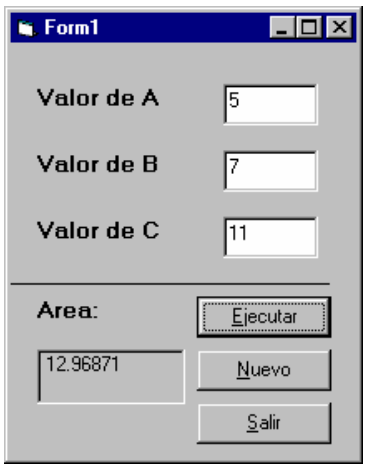
```
PRIVATE SUB CMDEJECUTAR_CLICK()
DIM A AS SINGLE, B AS SINGLE, C AS SINGLE
DIM IMAGINARIO AS BOOLEAN
DIM RAIZ1 AS SINGLE, RAIZ2 AS SINGLE, DISC AS SINGLE
DIM RESP AS INTEGER
A = VAL(INPUTBOX("VALOR A:", "ECUACIÓN"))
B = VAL(INPUTBOX("VALOR B:", "ECUACIÓN"))
C = VAL(INPUTBOX("VALOR C:", "ECUACIÓN"))
IF A = 0 AND B = 0 THEN
 MSGBOX "ECUACIÓN SIN SOLUCIÓN", VBCRITICAL, "ECUACIÓN"
 ELSE
  IF A = 0 THEN
   RAIZ1 = C / B * (-1)
   RAIZ2 = RAIZ1
   ELSE
    DISC = B ^ 2 - 4 * A * C
    IF DISC < 0 THEN
     IMAGINARIO = TRUE
     DISC = ABS(DISC)
    END IF
    RAIZ1 = (-B + SQR(DISC)) / (2 * A)
    RAIZ2 = (-B - SQR(DISC)) / (2 * A)
   END IF
  MSGBOX "RAIZ1=" & FORMAT(RAIZ1, "###0.0000"), VBINFORMATION, "ECUACIÓN"
  MSGBOX "RAIZ2=" & FORMAT(RAIZ2, "###0.0000"), VBINFORMATION, "ECUACIÓN"
   IF IMAGINARIO THEN
    MSGBOX "ES IMAGINARIO", VBEXCLAMATION
    END IF
END IF
RESP = MSGBOX("DESEA CONTINUAR:", VBYESNO + VBQUESTION, "ECUACIÓN")
IF RESP = VBYES THEN
 CMDEJECUTAR.SETFOCUS
ELSE
 END
END IF
END SUB
```

EJERCICIO Nº 10

Diseñar y codificar un programa que permita calcular el área de un triángulo en función del semiperímetro.

P=(a+b+c)/2

Donde: p= semiperímetro, A,b,c: Lados



$$Area = \sqrt{p(p-a)(p-b)(p-c)}$$

```
PUBLIC A AS SINGLE, B AS SINGLE, C AS SINGLE
PRIVATE SUB TXTA_KEYPRESS(KEYASCII AS INTEGER)
 A = VAL(TXTA.TEXT)
 IF KEYASCII = 13 THEN
   IF A > 0 THEN
    TXTB.SETFOCUS
END IF
END IF
END SUB

PRIVATE SUB CMDEJECUTAR_CLICK()
 DIM P AS SINGLE, AREA AS SINGLE
 IF (A + B) > C AND (A - B) < C THEN
  P = (A + B + C) / 2
  AREA = SQR(P * ABS(P - A) * ABS(P - B) * ABS(P - C))
  LBLAREA.CAPTION = STR(AREA)
  ELSE
   LBLAREA.CAPTION = "ERROR"
   CMDNUEVO.SETFOCUS
 END IF
END SUB
```

EJERCICIO Nº 11

Escribir un programa que determine si un año ingresado por teclado es bisiesto o no

Private Sub CmdNuevo_Click()
Txtaño.Text = ""
LblMsj.Caption = ""
Txtaño.SetFocus
End Sub

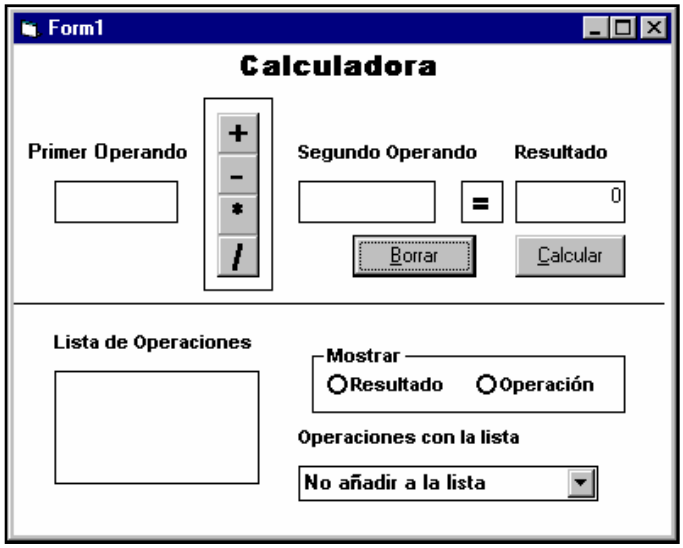
Private Sub Txtaño_KeyPress(KeyAscii As Integer)

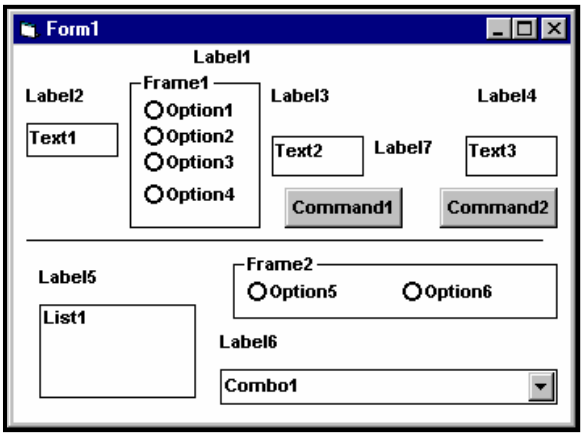
```
Dim Año As Integer
If KeyAscii = 13 Then
 Año = Val(Txtaño.Text)
If Txtaño.Text <> "" And Año <> 0 And Año > 0 Then
  If Año Mod 4 <> 0 Then
   LblMsj.Caption = "No es bisiesto"
  Else
   If Año Mod 100 <> 0 Then
    LblMsj.Caption = "Si es bisiesto"
   Else
    If Año Mod 400 = 0 Then
     LblMsj.Caption = "Si es bisiesto"
    Else
     LblMsj.Caption = "No es bisiesto"
    End If
   End If
  End If
  CmdNuevo.SetFocus
 End If
End If
End Sub

Private Sub Cmdsalir_Click()
End
End Sub
```

EJERCICIO Nº 12

Codificar en Visual Basic una pequeña calculadora de acuerdo al siguiente diseño

Solución:

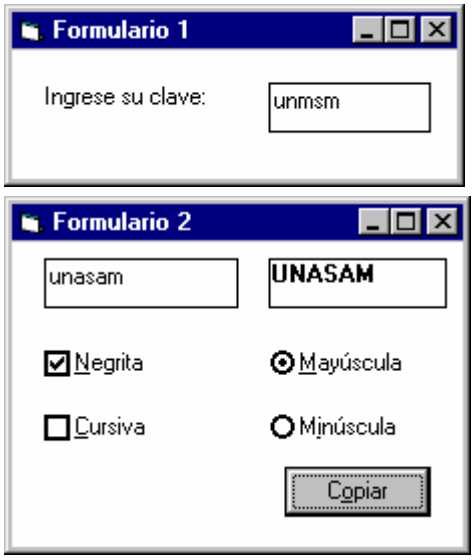




Control	Propiedad	Valor	Control	Propiedad	Valor
Label 1	Caption	Calculadora	Option1	Name	OptSum
Label 2	Caption	Primer Operando		Caption	+
Label 3	Caption	Segundo Operando		Style	Graphical
Label 4	Caption	Resultado		TooltipText	Bla, bla, bla,...
Label 5	Caption	Lista de Operaciones	Option2	Name	OptRes
Label 6	Caption	Operaciones con la Lista		Caption	-
Label 7	Caption	=	Option3	Name	OptMul
	Alignment	1-Center Justify		Caption	X
	BorderStyle	1-Fixed Single	Option4	Name	OptDiv
Text1	Name	TxtOp1		Caption	/
	Text	Nada	Option5	Name	OptResul
Text2	Name	TxtOp2		Caption	Resultado
	Text	Nada	Option6	Name	OptOpe
Text3	Name	TxtResul		Caption	Operación
	Text	0			
	Alignment	1-Rigth Justify	List1	Name	List1
	Locked	True	Combo1	Name	Cboañadir
	Multiline	True		List	Añadir lista
	Mouse Point	12-No Drop			No añadir lista
Command 1	Caption	&Borrar		Text	No añadir a la lista
	Name	CmdBorrar			
Command 2	Caption	&Calcular			
	Name	CmdCalcular			
Frame1	Caption	Nada			
Frame2	Caption	Mostrar			

```
PRIVATE SUB CMDCALCULAR_CLICK()
DIM OPE AS STRING
IF TXTOP1.TEXT = "" OR TXTOP2.TEXT = "" THEN
 MSGBOX "FALTAN DATOS"
 EXIT SUB
END IF
IF OPTSUM.VALUE = TRUE THEN TXTRESUL.TEXT = VAL(TXTOP1.TEXT) + VAL(TXTOP2.TEXT)
IF OPTRES.VALUE = TRUE THEN TXTRESUL.TEXT = VAL(TXTOP1.TEXT) - VAL(TXTOP2.TEXT)
IF OPTMUL.VALUE = TRUE THEN TXTRESUL.TEXT = VAL(TXTOP1.TEXT) * VAL(TXTOP2.TEXT)
IF OPTDIV.VALUE = TRUE THEN
 IF VAL(TXTOP2.TEXT) <> 0 THEN
  TXTRESUL.TEXT = VAL(TXTOP1.TEXT) / VAL(TXTOP2.TEXT)
 ELSE
  MSGBOX "DIVISIÓN POR CERO"
  EXIT SUB
 END IF
END IF
IF CBOAÑADIR.TEXT = "AÑADIR LISTA" THEN
 IF OPTRESUL.VALUE = TRUE THEN
  IF OPTSUM.VALUE = TRUE THEN OPE = "+"
  IF OPTRES.VALUE = TRUE THEN OPE = "-"
  IF OPTMUL.VALUE = TRUE THEN OPE = "*"
  IF OPTDIV.VALUE = TRUE THEN OPE = "/"
  LIST1.ADDITEM TXTOP1.TEXT & OPE & TXTOP2.TEXT & "=" & TXTRESUL.TEXT
 ELSE
  LIST1.ADDITEM TXTRESUL.TEXT
 END IF
 END IF
END SUB
```

```
PRIVATE SUB CMDBORRAR_CLICK()
TXTOP1.TEXT = ""
TXTOP2.TEXT = ""
TXTRESUL.TEXT = ""
OPTSUM.VALUE = FALSE
OPTRES.VALUE = FALSE
OPTMUL.VALUE = FALSE
OPTDIV.VALUE = FALSE
OPTOPE.VALUE = FALSE
OPTRESUL.VALUE = FALSE
LIST1.CLEAR
CBOAÑADIR.CLEAR
TXTOP1.SETFOCUS
END SUB
```



PROBLEMA PROPUESTO:

DISEÑAR Y CODIFICAR UN PROGRAMA QUE PERMITA CALCULAR LAS OPERACIONES BÁSICAS SUMAR Y RESTAR EN LAS SIGUIENTES BASES NUMÉRICAS: DECIMAL, BINARIO, OCTAL Y HEXADECIMAL.

EJERCICIO Nº 12

Diseñar un formulario que permita tener acceso a otro formulario, siempre y cuando ingrese la clave correcta; si no acierta con ella en tres oportunidades debe salir del programa.
El segundo formulario debe copiar un texto modificando el estilo de fuente y cambiando de mayúsculas a minúsculas y viceversa, eligiendo botones de chequeo (negrita y cursiva) y opciones (mayúscula y minúscula), respectivamente.

```
Private Sub Txtclave_KeyPress(KeyAscii As
Integer)
Dim Clave As String
If KeyAscii = 13 Then
Clave = UCase(Trim(Txtclave.Text))
  If Clave = "UNASAM" Then
    Form2.Show
  Else
    i = i + 1
    If i <> 3 Then
      MsgBox i & " No es su clave", vbExclamation
      Txtclave.Text = ""
     Else
      MsgBox "Lo siento", vbCritical
      End
    End If
  End If
End If
End Sub
```

Control	Propiedad	Valor
Label1	Caption	Ingrese su clave
Text1	Text	Nada
	Name	TxtClave

```
Private Sub CmdCopiar_Click()
   LblLetras.Caption = TxtLetras.Text
   If ChkNegrita.Value = 1 Then
      LblLetras.Font.Bold = True
   Else
      LblLetras.Font.Bold = False
   End If
   If ChkCursiva.Value = 1 Then
      LblLetras.Font.Italic = True
   Else
```

Control	Propiedad	Valor
Text1	Text	Nada
	Name	TxtLetras
Label1	Caption	Nada
	Name	LblLetras
	appearance	1-3D
Check1	Caption	Negrita
	Name	ChkNegrita
Check2	Caption	Cursiva
	Name	ChkCursiva
Option1	Caption	Mayúscula
	Name	OptMayuscula
Option2	Caption	Minúscula
	Name	OptMinúscula
Command1	Caption	Salir
	Name	Cmdsalir
Command2	Caption	Copiar
	Name	CmdCopiar

```
      LblLetras.Font.Italic = False
   End If
   If OptMayuscula.Value = True Then
      LblLetras.Caption = UCase(LblLetras.Caption)
   Else
      LblLetras.Caption = LCase(LblLetras.Caption)
   End If
End Sub
```

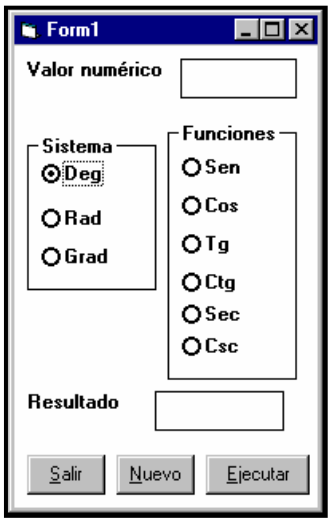
Problema:
Escribir un programa que permita convertir un número decimal en binario, octal y hexadecimal.

EJERCICIO Nº 13
DISEÑAR Y CODIFICAR UN PROGRAMA PARA CALCULAR LOS VALORES NUMÉRICOS DE LAS FUNCIONES TRIGONOMÉTRICAS EN EL SISTEMA RADIÁN, SEXAGESIMAL Y CENTESIMAL.
SOLUCIÓN:

CONTROL	PROPIEDAD	Valor
TEXT1	NAME	TXTN
FRAME1	CAPTION	SISTEMA
OPTION1	NAME	OPTDEG
OPTION2	NAME	OPTRAD
OPTION3	NAME	OPTGRAD
FRAME2	CAPTION	FUNCIONES
OPTION4	NAME	OPTSEN
	CAPTION	SEN
	STYLE	GRAPHICAL
	TOOLTIPTEXT	BLA, BLA, BLA,
OPTION5	NAME	OPTCOS
OPTION6	NAME	OPTTG
OPTION7	NAME	OPTCTG
OPTION8	NAME	OPTSEC
OPTION9	NAME	OPTCSC
TEXT2	NAME	TXTRESUL

```
PRIVATE SUB CMDEJECUTAR_CLICK()
DIM S AS SINGLE, R AS SINGLE, SEN AS SINGLE DIM COSE AS SINGLE, TG AS SINGLE
DIM CTG AS SINGLE, SEC AS SINGLE
DIM CSC AS SINGLE
DIM C AS SINGLE
CONST PI = 3.14159265359
IF OPTDEG.VALUE = TRUE THEN
 S = VAL(TXTN.TEXT)
 R = PI * S / 180
END IF
'COPY LEFT. SALAZAR CÓRDOVA, HÉCTOR

IF OPTRAD.VALUE = TRUE THEN
R = VAL(TXTN.TEXT)
END IF
IF OPTGRAD.VALUE = TRUE THEN
C = VAL(TXTN.TEXT)
R = PI * C / 200
END IF
SEN = VAL(FORMAT(SIN(R), "###0.00000"))
COSE = VAL(FORMAT(COS(R), "###0.00000"))
IF OPTSEN.VALUE = TRUE THEN TXTRESUL.TEXT = SEN
IF OPTCOS.VALUE = TRUE THEN TXTRESUL.TEXT = COSE
IF OPTTG.VALUE = TRUE THEN
  IF COSE <> 0 THEN
    TG = SEN / COSE
```

```
    TXTRESUL.TEXT = STR(TG)
   ELSE
    TXTRESUL.TEXT = "INDEFINIDO"
  END IF
END IF
IF OPTCTG.VALUE = TRUE THEN
  IF SEN <> 0 THEN
    CTG = COSE / SEN
    TXTRESUL.TEXT = STR(CTG)
   ELSE
    TXTRESUL.TEXT = "INDEFINIDO"
  END IF
END IF
IF OPTSEC.VALUE = TRUE THEN
  IF COSE <> 0 THEN
    SEC = 1 / COSE
    TXTRESUL.TEXT = STR(SEC)
   ELSE
    TXTRESUL.TEXT = "INDEFINIDO"
  END IF
END IF
IF OPTCSC.VALUE = TRUE THEN
 IF SEN <> 0 THEN
    CSC = 1 / SEN
    TXTRESUL.TEXT = STR(CSC)
   ELSE
            TxtResul.Text = "Indefinido"
  END IF
END IF
END SUB
PRIVATE SUB CMDNUEVO_CLICK()
TXTN.TEXT = ""
         OptSen.Value = False
OPTCOS.VALUE = FALSE
OPTTG.VALUE = FALSE
OPTCTG.VALUE = FALSE
OPTSEC.VALUE = FALSE
OPTCSC.VALUE = FALSE
TXTRESUL.TEXT = ""
TXTN.SETFOCUS
END SUB

PRIVATE SUB CMDSALIR_CLICK()
 UNLOAD ME
 END
END SUB
```



EJERCICIO Nº 14

Crear una aplicación que encienda y apague una bombilla al mismo tiempo que en un botón de comando el mensaje que aparece cambia a ON y OFF. Se debe añadir un botón con el mensaje Salir.
El formulario a diseñar deberá tener los siguientes controles con las propiedades:

CONTROL	PROPIEDAD	VALOR
Command Button	Caption	Salir
	Name	cmdSalir
Command Button	Caption	ON
	Name	cmdOnOff
Picture Box	BorderStyle	0-None
	Name	picOff

	Picture	vb\icons\misc\Lightoff.ico
Picture Box	BorderStyle	0-None
	Name	PicOn
	Picture	vb\icons\misc\Lighton.ico

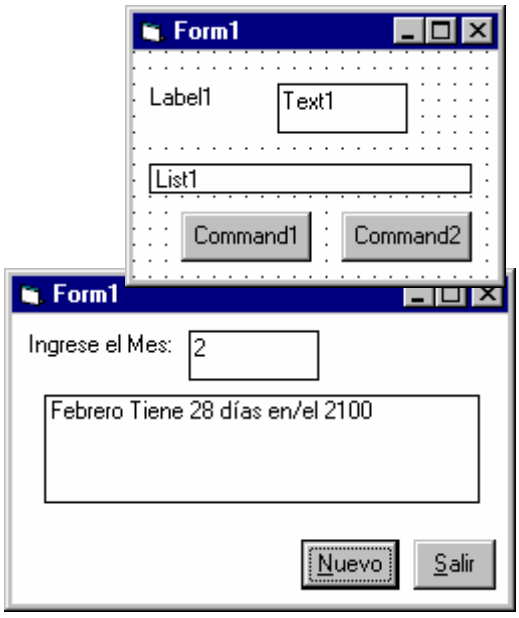
El código a escribir será:

```
Private Sub Form_Initialize()
‘Centrar el formulario
Left = (Screen.Width - F.Width) / 2
Top = (Screen.Height - F.Height) / 2
‘Posicionar las linternas en la misma posición
picOn.Left = 1200
picOn.Top = 600
picOff.Left = 1200
picOff.Top = 600
‘Asegurarse que el programa empieza con la linterna apagada
picOff.Zorder 0
End Sub
Private Sub cmdClose_Click()
‘Finalizar la aplicación
Unload cmdClose.Parent
End Sub
Private Sub cmdOnOff_Click()
‘Cambiar la linterna de ON a OFF o viceversa
If cmdOnOff.Caption = "ON" Then
cmdOnOff.Caption = "OFF"
‘Traer la linterna encendida delante
picOn.ZOrder 0
Else
cmdOnOff.Caption = "ON"
‘Traer la linterna apagada delante
picOff.ZOrder 0
End If
End Sub
```

EJERCICIO Nº 15

Elaborar un programa que determine la cantidad de días que tiene un determinado mes. Tener en cuenta si el mes de febrero pertenece a un año bisiesto.

```
Private Sub TxtMes_KeyPress(KeyAscii As Integer)
Dim Mes As Integer, año As Integer
Dim Nommes As String
If KeyAscii = 13 Then
Mes = Val(TxtMes.Text)
If Mes >= 1 And Mes <= 12 Then
Select Case Mes
 Case 1   Nommes = "Enero"
 Case 2   Nommes = "Febrero"
 Case 3   Nommes = "Marzo"

 Case 4   Nommes = "Abril"
 Case 5   Nommes = "Mayo"
 Case 6   Nommes = "Junio"
 Case 7   Nommes = "Julio"
 Case 8   Nommes = "Agosto"
 Case 9   Nommes = "Setiembre"
 Case 10  Nommes = "Octubre"
 Case 11  Nommes = "Noviembre"
 Case 12  Nommes = "Diciembre"
 End Select
 Select Case Mes
```



```
  Case 2
   año = Val(InputBox("Ingres el año de referencia:"))
  If año > 0 Then
   If año Mod 4 <> 0 Then
    MsgBox Nommes & " Tiene 28 días", vbInformation
    List1.AddItem Nommes & " Tiene 28 días" & " en/el " & año
   Else
    If año Mod 100 <> 0 Then
     MsgBox Nommes & " Tiene 29 días", vbInformation
     List1.AddItem Nommes & " Tiene 29 días" & " en/el " & año
    Else
     If año Mod 400 = 0 Then
      MsgBox Nommes & " Tiene 29 días", vbInformation
      List1.AddItem Nommes & " Tiene 29 días" & " en/el " & año
     Else
      MsgBox Nommes & " Tiene 28 días", vbInformation
      List1.AddItem Nommes & " Tiene 28 días" & " en/el " & año
     End If
    End If
   End If
 End If
 Case 1, 3, 5, 7, 8, 10, 12
   MsgBox Nommes & " Tiene 31 días", vbInformation
   List1.AddItem Nommes & " Tiene 31 días"
 Case 4, 6, 9, 11
   MsgBox Nommes & " Tiene 30 días", vbInformation
   List1.AddItem Nommes & " Tiene 30 días"
End Select
CmdNuevo.SetFocus
End If
End If
End Sub
```

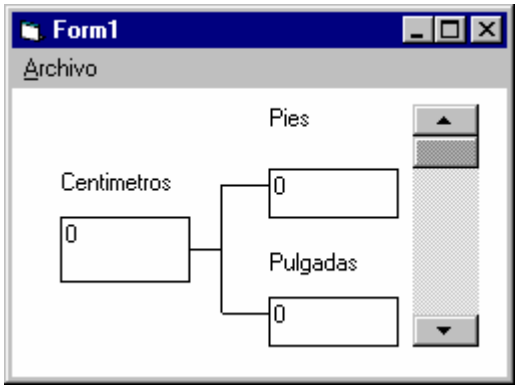
EJERCICIO Nº 16

Problema:

Elaborar un programa que permita convertir medidas de longitud. Centímetros a pies y pulgadas.

Control	Propiedades	Valor	Control	Propiedad	Valor
Form1	Name	Frmlongitud	Text3	Name	TxtPulg
		Conversor de longitudes		Text	Pulgadas
Menú	Name	Mnuarchivo	Vscroll1	Name	VsMedida
	Caption	&Archivo		Min	0
Submenú	Name	MnuArchivoexit		Max	250
	Caption	E&xit		Smallchange	1
Text1	Name	TxtCm		LargeChange	10
	Text	0	Label1	Caption	Centímetros
Text2	Name	TxtPies	Label2	Caption	Pies
	Text	0	Label3	Caption	Pulgadas

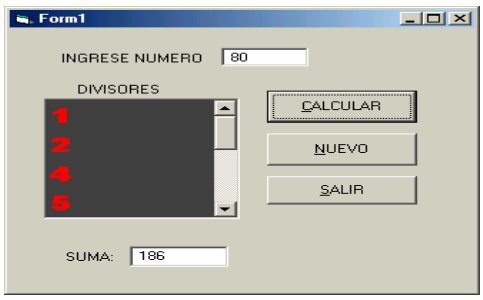
```
Private Sub VSmedida_Change()
 Txtcm.Text = VSmedida.Value
 txtpies.Text = 1 / 100 * 3.280084 * VSmedida.Value
 Txtcm.Text = VSmedida.Value
 Txtpulg.Text = 0.8937008 * VSmedida.Value
End Sub

Private Sub MnuArchivoExit_Click()
 Beep
 End
End Sub
```



HScrollBar, VScrollBar (controles)
Las barras de desplazamiento permiten explorar fácilmente una larga lista de elementos o una gran cantidad de información. Cuando utiliza una barra de desplazamiento como indicador de cantidad o velocidad o como dispositivo de entrada, utilice las propiedades Max y Min para establecer el rango adecuado del control.
Para especificar la cantidad de cambio que va a iniciar la barra de desplazamiento, utilice la propiedad LargeChange para hacer clic en la barra de desplazamiento, y la propiedad SmallChange para hacer clic en las flechas de los extremos de la barra. La propiedad Value de la barra de desplazamiento aumenta o disminuye los valores establecidos para las propiedades LargeChange y SmallChange. Puede colocar el cuadro de desplazamiento en tiempo de ejecución estableciendo Value entre 0 y 32.767, inclusive.

Ejercicio: Elaborar un programa que permita convertir grados centígrados en grados fahrenheit, utiilizando Hscrollbar.

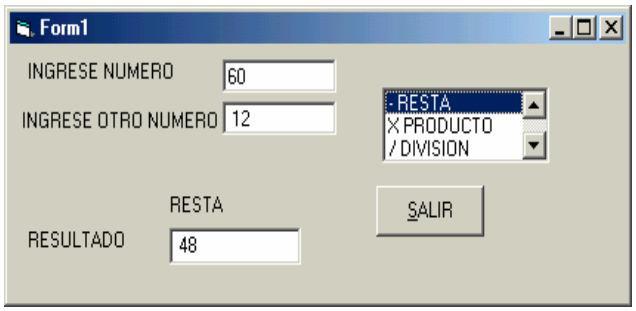
EJERCICIO Nº 17
Elaborar un programa para calcular los números divisores de un numero ingresado por teclado




```
Private Sub Command1_Click()
Dim i, n, d As Integer
n = Val(Text1.Text)
For i = 1 To n
   If n Mod i = 0 Then
      List1.AddItem i
      s = s + i
    End If
Next i
Text2.Text = Str$(s)
End Sub
```

```
Private Sub Command2_Click()
Text1.Text = ""
Text1.Text = ""
List1.Clear
Text1.SetFocus
End Sub
Private Sub Command3_Click()
Unload Me
End Sub
```

EJERCICIO Nº 18
Crear un programa en Visual Basic para que usando un cuadro de lista y dos números se calcule la operación correspondiente seleccionada. Mejorar este programa insertando el botón nuevo.



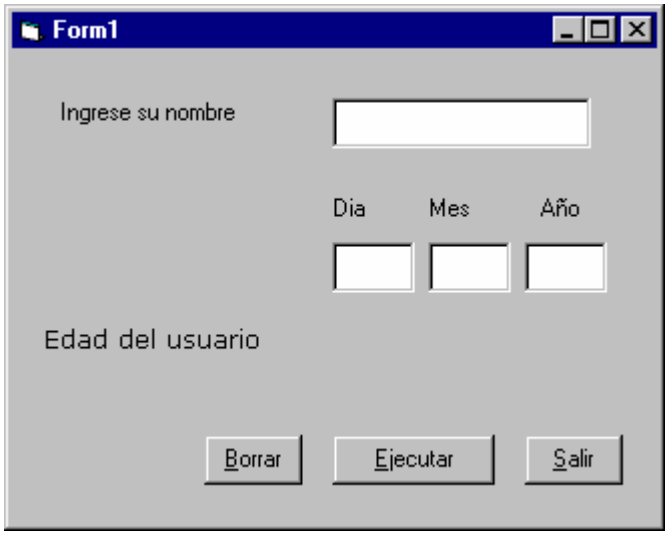

```
Private Sub Form_Load()
Dim OPER(0 To 3) As String
OPER(0) = "+ SUMA"
OPER(1) = "- RESTA"
OPER(2) = "X PRODUCTO"
OPER(3) = "/ DIVISION"
For I = 0 To 3
  List1.AddItem OPER(I)
Next I
End Sub
Private Sub Command1_Click()
Unload Me
End Sub
```

```
Private Sub List1_Click()
Dim N1, N2, S As Double
N1 = Val(Text1.Text)
N2 = Val(Text2.Text)
If List1.ListIndex = 0 Then
   Text3.Text = Str$(N1 + N2)
   Label4.Caption = "SUMA"
End If
If List1.ListIndex = 1 Then
   Text3.Text = Str$(N1 - N2)
   Label4.Caption = "RESTA"
End If
If List1.ListIndex = 2 Then
   Text3.Text = Str$(N1 * N2)
   Label4.Caption = "PRODUCTO"
End If
If List1.ListIndex = 3 Then
   Text3.Text = Str$(N1 / N2)
   Label4.Caption = "DIVISION"
End If
End Sub
```

EJERCICIO Nº 19

Elaborar un programa que permita calcular la edad de un usuario.

```
Private Sub cmdejecutar_Click()
Dim Fnac As String
Dim edad As Single
Dim Fn As Date
Fn = CDate(Txtfnac.Text)
edad = Int((Date - Fn) / 365.25)
lblresp.Caption = edad
End Sub
```



```
Private Sub cmdejecutar_Click()
Dim Nom As String, D As String * 2, M As String * 2, A As String * 4
Dim Fn As Date
Dim edad As Single
Nom = txtnom.Text
D = TxtD.Text
M = TxtM.Text
A = TxtA.Text
Fn = CDate(D & "/" & M & "/" & A)
edad = Int((Date - Fn) / 365.25)
lblresp.Caption = "Estimado/da " & Nom & " usted tiene" & Str(edad) & " Años"
End Sub
```

EJERCICIO Nº 20

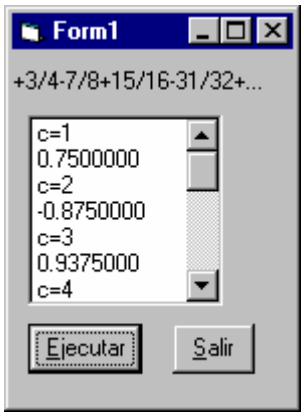
Elaborar un programa que permita ingresar por teclado una serie de notas que se visualicen en una lista. Se debe calcular el promedio

```
Private Sub TxtN_KeyPress(KeyAscii As Integer)
Dim Resp As Integer
If KeyAscii = 13 Then
 i = i + 1
 n = Val(TxtN.Text)
 List1.AddItem i & Space(10) & Str(n)
 TxtN.SetFocus
 TxtN.Text = ""
  s = s + n
 Resp = MsgBox("Desea ingresar otro dato:", vbInformation +
vbYesNo + vbDefaultButton1)
 If Resp = vbNo Then
   CmdCalcular.SetFocus
 End If
End If
End Sub
```


```
Private Sub CmdNuevo_Click()
TxtN.Text = ""
txtp.Text = ""
List1.Clear
TxtN.SetFocus
End Sub

Private Sub CmdSalir_Click()
Unload Me
End
End Sub
```

EJERCICIO Nº 21

```
Private Sub CmdEjecutar_Click()
'Mostrar los N primeros términos de la siguiente serie
'indicando ademas la suma de los mismos:
' 7; 9; 12; 16; 21; ...
Dim I As Single, t As Single, c As Single
Dim N As Single
N = Val(TxtNrot.Text)
I = 1
t = 7
Do
List1.AddItem Str(t)
I = I + 1
c = c + 1
t = t + I
Loop Until c = N
End Sub
```



EJERCICIO Nº 22

```
Private Sub CmdEjecutar_Click()
Dim i As Single, j As Single, n As Single
Dim d As Single, t As Single, s As Single
Dim c As Single
c = 0
i = 0
j = 1
n = 1
s = 0
Do
```

```
c = c + 1
i = i + 1
j = j + 1
n = n + 2 ^ i
d = 2 ^ j
List1.AddItem "c=" & c
If j Mod 2 = 0 Then
t = n / d
s = s + t
List1.AddItem Format(t, "###0.0000000")
Else
t = -n / d
s = s + t
List1.AddItem Format(t, "###0.0000000")
End If
Loop Until c = 10
List1.AddItem Format(s, "###0.0000")
End Sub

Private Sub CmdSalir_Click()
 Beep
 End
End Sub
```

EJERCICIO Nº 23

Elaborar un programa que permita calcular el costo de alquiler de una computadora teniendo como datos la hora de entrada y la hora de salida. La hora de alquiler es de S/. 1.50


```
Private Sub CmdEjecutar_Click()
 Dim He As Date, Hs As Date, h As Single, tt As Date
 Dim the As Single, ths As Single, mt As Single
 Dim Costo As Single
 He = CDate(Txthe.Text)
 the = Hour(He) * 60 + Minute(He)
 Hs = CDate(Txths.Text)
 ths = Hour(Hs) * 60 + Minute(Hs)
 tt = Hs - He
 LblTT.Caption = tt
 mt = ths - the
 Costo = 1.5 * mt / 60
```

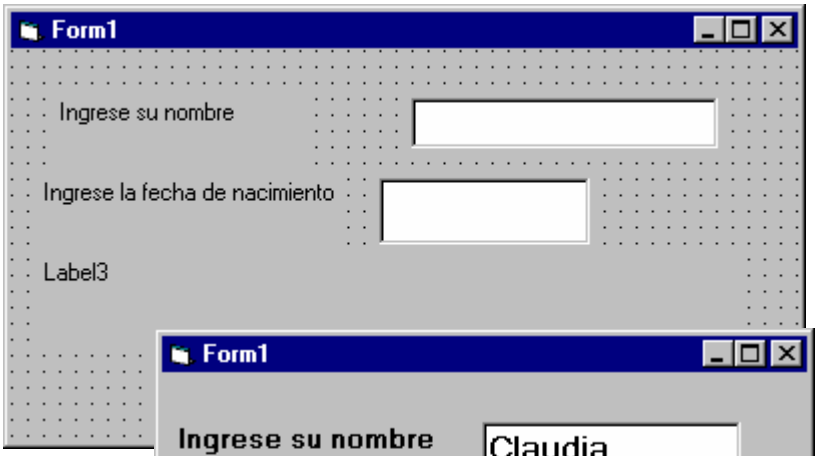
```
LblCosto.Caption = Format(Costo, "###0.00")
End Sub
```

EJERCICIO Nº 24

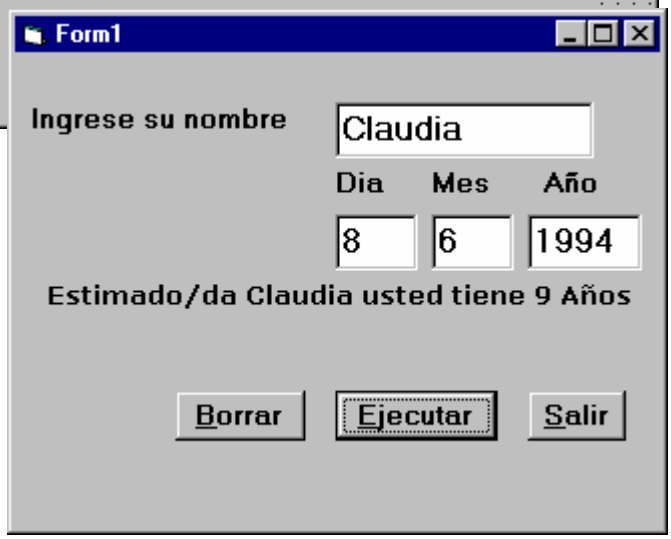
Elaborar un programa que permita calcular la edad de un usuario. El programa debe leer por teclado la fecha dd/mm/aa

Sol.

```
Private Sub cmdejecutar_Click()
Dim Fnac As String
Dim edad As Single
Dim Fn As Date
Fn = CDate(Txtfnac.Text)
edad = Int((Date - Fn) / 365.25)
lblresp.Caption = edad
End Sub
```



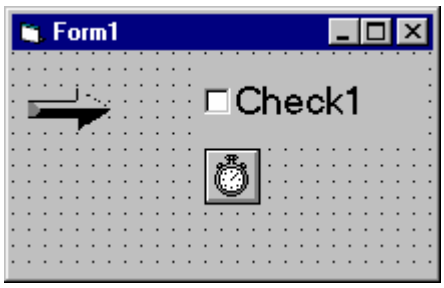

```
Private Sub CmdBorrar_Click()
txtnom.Text = ""
TxtD.Text = ""
TxtM.Text = ""
TxtA.Text = ""
lblresp.Caption = "Edad del usuario"
txtnom.SetFocus
End Sub
```




```
Private Sub cmdejecutar_Click()
Dim Nom As String, D As String * 2, M As String
* 2, A As String * 4
Dim Fn As Date
Dim edad As Single
Nom = txtnom.Text
D = TxtD.Text
M = TxtM.Text
A = TxtA.Text
Fn = CDate(D & "/" & M & "/" & A)
edad = Int((Date - Fn) / 365.25)
lblresp.Caption = "Estimado/da " & Nom & " usted tiene" & Str(edad) & " Años"
End Sub
```

EJERCICIO Nº 25

En este ejemplo se carga un icono flecha de un directorio de iconos en un control Image control. La flecha se arrastra por el formulario cuando la propiedad Stretch tiene el valor True, y salta el formulario cuando el valor es False. Para probar este ejemplo, pegue el código en la sección de declaraciones de un formulario que contenga un control Image, un control CheckBox y un control Timer, presione F5 y haga clic en el formulario. Asegúrese de comprobar la ruta de acceso al directorio de iconos y cambiarla si es necesario. Para ver los distintos efectos de la propiedad Stretch
, haga clic en el control CheckBox, y luego en el formulario de nuevo.




```
Option Explicit
Dim AnImg    'Declara variable.

Private Sub Form_Click()
Crono1.Enabled = True    'Habilita el cronómetro.
End Sub

Private Sub Form_Load()
'Carga un icono en el control Image.
Imagen1.Picture = LoadPicture("C:\Flecha.ICO")
Imagen1.Left = 0     'Mueve la imagen al borde izquierdo.
AnImg = Imagen1.Width    'Guarda la anchura de la imagen.
Crono1.Interval = 300
Crono1.Enabled = False   'Deshabilita el cronómetro.
Verif1.Caption = "Propiedad strech"
End Sub

Private Sub Crono1_Timer()
Static MueveIcono As Integer     'Indicador para desplazar el icono.
If Not MueveIcono Then
Imagen1.Move Imagen1.Left + AnImg, Imagen1.Top, AnImg * 2
Else
 'Mueve la imagen y le devuelve su anchura original.
Imagen1.Move Imagen1.Left + AnImg, Imagen1.Top, AnImg
End If
 'Si la imagen está fuera del borde del formulario, empezar otra vez.
If Imagen1.Left > ScaleWidth Then
Imagen1.Left = 0
Crono1.Enabled = False
End If
 'MueveIcono = Not MueveIcono  Restablece el indicador.
End Sub

Private Sub Verif1_Click()
Imagen1.Stretch = Verif1.Value
End Sub
```

Ejercicio 26

Escribir un programa que permita calcular el peso ideal de una persona, según la fórmula de Lorentz.
Para Varones PI=(E-100) - (E-150)/4
Para Mujeres el 5% menos del varón.

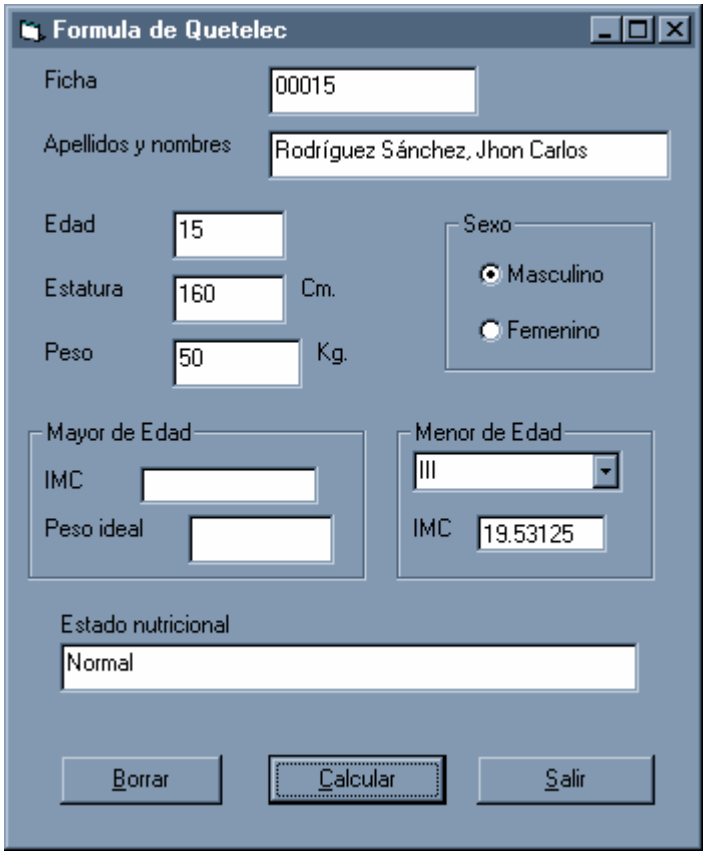

```
Private Sub CmdBorrar_Click()
 cbosexo.Text = "varón"
 TxtE.Text = ""
 TxtP.Text = ""
 TxtP.Locked = True
 cbosexo.SetFocus
End Sub

Private Sub CmdCalcular_Click()
 Dim p As Single, e As Single
 e = Val(TxtE.Text)
 If TxtE.Text = "" Then
  MsgBox ("Falta datos")
  Exit Sub
 End If
 If cbosexo.Text = "varón" Then
  p = (e - 100) - (e - 150) / 4
  TxtP.Text = p
End If
If cbosexo.Text = "mujer" Then
  p = (e - 100) - (e - 150) / 4
  p = p - 5 / 100 * p
  TxtP.Text = p
End If
End Sub

Private Sub Form_Load()
 TxtP.Locked = True
End Sub
```

Ejercicio 27.
Diseñar un programa que permita calcular el índice de masa corporal, peso ideal y estado nutricional de un paciente aplicando la teoría de Tanner según el peso y la edad.

Tabla de IMC según el sexo y la edad. Para adolescentes desde los 12 hasta los 18 años

Grado	Mujeres	Varones
I	[15.2 18]	[15.4 18.1]
II	[15.9 19]	[16.3 18.9]
III	[16.9 20]	[17.1 19.6]
IV	[18.1 21.4]	[18 20.5]
V	[19.2 23.2]	[18.6 21.4]

Fórmulas necesarias:
IMC=Peso/Estatura²
Varón---> Peso ideal=Estatura-100 -(Estatura-150)/4
Mujer--->Peso ideal=Peso ideal -5% del Peso ideal




```
Private Sub CmdCalcular_Click()
Dim Edad As Single, P As Single, E As Single
Dim IMC As Single, Pi As Single
 Edad = Val(TxtEdad.Text)
 P = Val(TxtP.Text)
 E = Val(TxtE.Text)
 IMC = P / (E / 100) ^ 2
 Pi = (E - 100) - (E - 150) / 4
 If Edad < 10 Or Edad > 50 Then
  MsgBox ("Edad no prevista")
  Exit Sub
```

```
End If
If Edad >= 18 And Edad <= 50 Then
  If OptM.Value = True Then
    TxtIMCMa.Text = IMC
    TxtPI.Text = Pi
  End If
  If OptF.Value = True Then
    TxtIMCMa.Text = IMC
     TxtPI.Text = Pi - 5 / 100 * Pi
End If
If IMC < 20 Then TxtEstado.Text = "Posible desnutrición"
If IMC >= 20 And IMC <= 25 Then TxtEstado.Text = "Normal"
If IMC > 25 And IMC <= 30 Then TxtEstado.Text = "Sobre peso"
If IMC > 30 And IMC<=40  Then TxtEstado.Text = "Obesidad"
If IMC > 40 Then TxtEstado.Text = "Obesidad morbida"
End If
If Edad >= 10 And Edad < 18 Then
 TxtIMCMe.Text = IMC
If OptF.Value = True Then
 Select Case CboGrado.Text
   Case "I"
      If IMC < 15.2 Then TxtEstado.Text = "Déficit"
      If IMC >= 15.2 And IMC < 18 Then TxtEstado.Text = "Normal"
      If IMC >= 18 Then TxtEstado.Text = "Sobrepeso"
  Case "II"
     If IMC < 15.9 Then TxtEstado.Text = "Déficit"
     If IMC >= 15.9 And IMC < 19 Then TxtEstado.Text = "Normal"
     If IMC >= 19 Then TxtEstado.Text = "Sobrepeso"
 Case "III"
       If IMC < 16.9 Then TxtEstado.Text = "Déficit"
       If IMC >= 16.9 And IMC < 20 Then TxtEstado.Text = "Normal"
       If IMC >= 20 Then TxtEstado.Text = "Sobrepeso"
 Case "IV"
      If IMC < 18.1 Then TxtEstado.Text = "Déficit"
      If IMC >= 18.1 And IMC < 21.4 Then TxtEstado.Text = "Normal"
      If IMC >= 21.4 Then TxtEstado.Text = "Sobrepeso"
  Case "V"
      If IMC < 19.2 Then TxtEstado.Text = "Déficit"
      If IMC >= 19.2 And IMC < 23.2 Then TxtEstado.Text = "Normal"
      If IMC >= 23.2 Then TxtEstado.Text = "Sobrepeso"
 End Select
 End If
  If OptM.Value = True Then
 Select Case CboGrado.Text
   Case "I"
      If IMC < 15.4 Then TxtEstado.Text = "Déficit"
      If IMC >= 15.4 And IMC < 18.1 Then TxtEstado.Text = "Normal"
      If IMC >= 18.1 Then TxtEstado.Text = "Sobrepeso"
  Case "II"
     If IMC < 16.3 Then TxtEstado.Text = "Déficit"
     If IMC >= 16.3 And IMC < 18.9 Then TxtEstado.Text = "Normal"
```

```
      If IMC >= 18.9 Then TxtEstado.Text = "Sobrepeso"
  Case "III"
       If IMC < 17.1 Then TxtEstado.Text = "Déficit"
       If IMC >= 17.1 And IMC < 19.6 Then TxtEstado.Text = "Normal"
       If IMC >= 19.6 Then TxtEstado.Text = "Sobrepeso"
  Case "IV"
      If IMC < 18 Then TxtEstado.Text = "Déficit"
      If IMC >= 18 And IMC < 20.5 Then TxtEstado.Text = "Normal"
      If IMC >= 20.5 Then TxtEstado.Text = "Sobrepeso"
   Case "V"
      If IMC < 18.6 Then TxtEstado.Text = "Déficit"
      If IMC >= 18.6 And IMC < 21.4 Then TxtEstado.Text = "Normal"
      If IMC >= 21.4 Then TxtEstado.Text = "Sobrepeso"
   End Select
   End If

 End If
End Sub


Private Sub TxtE_KeyPress(KeyAscii As Integer)
 If KeyAscii = 13 Then
   TxtP.SetFocus
 End If
End Sub


Private Sub TxtEdad_KeyPress(KeyAscii As Integer)
 If KeyAscii = 13 Then
   TxtE.SetFocus
 End If
End Sub
```