“PORTAFOLIO DE EVIDENCIAS”

APLICA LA METODOLOGIA DE DESARROLLO RAPIDO DE APLICACIONES CON PROGRAMACION ORIENTADA A EVENTOS

Profesor: L.I. Mario Lagunas Brito

NOMBRE: Luis Eduardo Bahena Castillo

GRADO Y GRUPO: 3BPM

Nº DE LISTA: #2

FECHA: VIERNES DE DEL 2019

CORREO ELECTRONICO:
luiseduardobahenacastillo007@gmail.com

INDICE

Conversiones de expresiones matemáticas a expresiones computacionales- Ejercicios

OPERADORES MAMEMATICOS	OPERADORES VISUAL BASIC
SUMA Y RESTA (+, -)	+, -
MULTIPLICACION (·, (), ab)	*
DIVISION ($\frac{a}{b}$, a/b, a÷b)	/, \, Mod
EXPONENTE (a^n)	a^n
RAIZ ($\sqrt[n]{a}$)	a^(1/n)

1. a^2b = a^2*b

2. (a+b) $(ab)^2$= (a+b)*a*b^2

3. $\frac{a}{b} + \frac{b}{c}$ = (a/b)+(b/c)

4. $\sqrt{ab + c^2}$ = (a*b+(c^2))^(1/2)

5. $\sqrt[3]{a - (bc)}$ = (a-(b*c))^(1/3)

6. 3ab + a^2= 3*(a*b)+a^2

7. $\frac{4x^2y}{2a}$= 4*(x^2)*y/2*a

8. $\sqrt[4]{\frac{ab}{b}}$= (a*b/b)^(1/4)

9. $\frac{4a^3b^2+c}{2bc}$= 4*(a^3)*(b^2)+c/2*b*c

10. $\frac{4ab}{\sqrt{b^2}}$= 4*(a*b)/(b^2)^(1/2)

Apunte en clase tema: Select Case y sentencias If

Sirven para que el programa pueda tomar decisiones, basado en una expresión relacional (>, <, >=, <=, <>, =, =!) o lógica (AND, NOT, OR)

Existen cuatro tipos de estas sentencias:

- Decision Simple (**If……Then**)
- Decision Doble (**If……Then….Else**)
- Decision Anidada
- Decisión Multiple (**Select…..Case**)

SENTENCIAS MULTIPLES

Evalua el valor de una variable entera, y dependiendo de su valor, realizara una serie de acciones. Solo se puede utilizar el operador de igualdad (=).
Se utiliza la sentencia select para hacer la selección multiple. Su sintaxis es:

```
Select Case Variable
    Case Valor 1
        Sentencias
    Case Valor 2
        Sentencias
    Case Valor n
        Sentencias
End Select
```

Se necesita conocer el rango de valores que puede tomar la variable.

Ejemplo:

```
Select Case Val(txtPrmdio.text)
    Case 10
        lbl.Mnsje.Caption = “Excelente”
    Case 9
        lbl.Mnsje.Caption = “Muy bien”
    Case 8
        lbl.Mnsje.Caption = “Bien”
    Case 7
        lbl.Mnsje.Caption = “Regular”
    Case 6
        lbl.Mnsje.Caption = “Suficiente”
    Case 5,4,3,2,1,0
        lbl.Mnsje.Caption = “Reprobado”
End Select
```

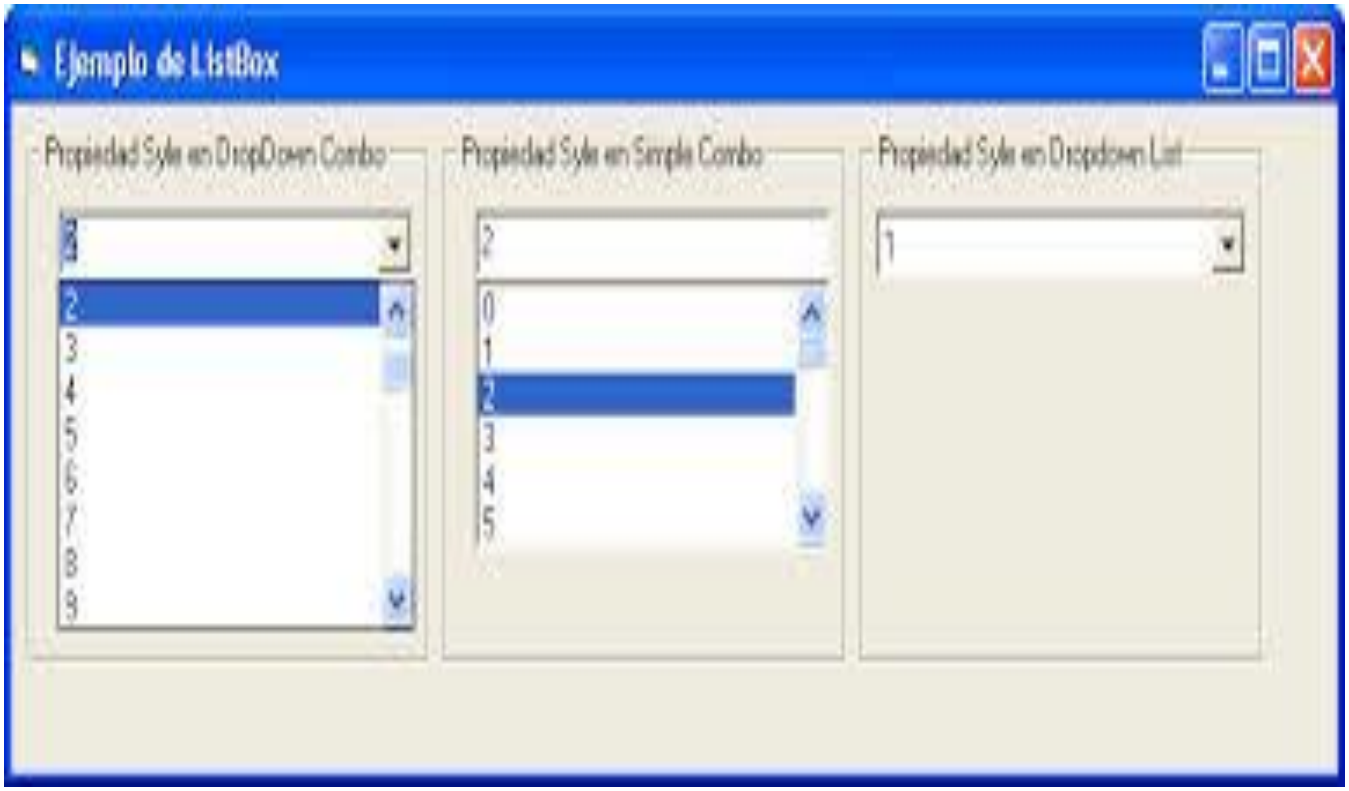
Apunte en clase tema: Controles ComboBox y ListBox

Permiten mostrar una lista de valores numéricos o de cadena, dichos valores se pueden agregar en tiempo de diseño o en tiempo de ejecución (código), sin embargo para eliminar elementos a la lista se puede hace solo mediante código.

Agregar Elementos: En tiempo de diseño se usa la propiedad ***List***. En tiempo de ejecución se utiliza el método ***.AddItem***

Eliminar Elementos: En tiempo de ejecución se usa el método ***.RemoveItem***

Cada elemento en la lista se identifica por un índice (como un arreglo) en cual se almacena en la propiedad ***.ListIndex***



Apunte enviado por correo: Colores en Visual Basic

El Color en Visual Basic 6.0

El color en Visual Basic 6.0 está definido por el sistema *R.G.B.* RGB proviene de las iniciales de Red-Green-Blue, y cada valor de cada color ocupa 1 byte (de 0 a 255), expresado en sistema hexagesimal (0 a 255, 00 a FF). El color ocupa 4 byes en la memoria: 1 para cada color (1 para Rojo, 1 para Verde y 1 para Azul) y un byte extra.
Los distintos colores surgen de los valores dados a cada una de las componentes de RGB, o sea que la cantidad de colores que se pueden formar es muy amplia.

Constantes de color.

Visual Basic 6.0 dispone de una serie de constantes de colores, al grupo de estas constantes se lo denomina *ColorConstants* y contiene las constantes que se encuentran en la tabla siguiente:

Constantes de Colores		
Color	**Constante**	**Valor**
Negro	vbBlack	0
Blanco	vbWhite	16777215
Amarillo	vbYellow	65535
Verde	vbGreen	65280
Rojo	vbRed	255
Azul	vbBlue	16711680
Turquesa	vbCyan	16776960
Magenta	vbMagenta	16711935

Estas constantes son algunos de los colores más usuales del sistema RGB, sin embargo es bastante limitado ya que con el sistema RGB se pueden obtener millones de combinaciones y por lo tanto millones de colores.

Función RGB.

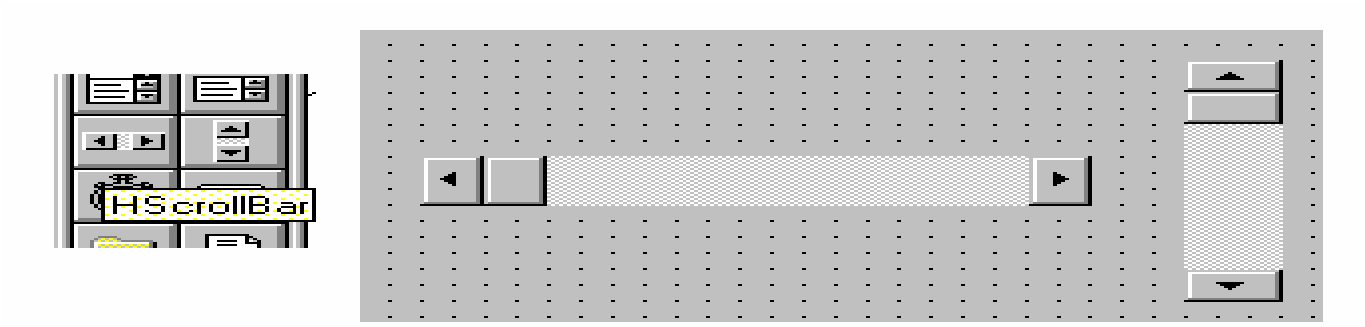
Existe una función con la cual obtener en cualquier color del sistema RGB y es mediante la función RGB. Esta función tiene por argumento los tres valores de las componentes R.G.B., que pueden tomar un valor de 0 a 255. Si se le da un valor mayor que 255, se toma el valor 255.
La forma general de la función RGB es la siguiente:

```
RGB(rojo as Integer,verde as Integer, azul as Integer)
```

Investigación: Controles Timer y ScrollBar con sus principales propiedades

HScrollBar y VScrollBar

Son dos controles similares, para introducir un dato cuasi-analógico en una aplicación. Se
toman directamente de la caja de herramientas, y tienen un aspecto parecido al de un control
de volumen de un equipo de música. El HScrollBar está en posición horizontal, y el VScrollBar
en posición vertical.



Mediante estos controles se pueden introducir datos variando la posición del cursor.

PROPIEDADES de HScrollBar y VScrollBar

Las señaladas con (*) son comunes a ambos controles y no presentan novedades respecto a las ya comentadas para los controles precedentes.

DragIcon (*)
DragMode (*)
Enabled (*)
Height (*)
HelpContextID (*)
Index (*)

LargeChange

Esta propiedad establece la variación de la propiedad Value cada vez que se hace click en el interior de la barra de desplazamiento, en la parte por donde pasa el cursor.
Left (*)

Max

Esta propiedad establece el valor máximo para la propiedad Value, es decir, el

valor de esta propiedad cuando el cursor está en su parte máxima. (Recuerde que el cursor está en el máximo, cuando está mas a la derecha, caso del HScrollBar, o cuando está en la parte mas baja, caso del HScrollBar.

Min
Esta propiedad establece el valor mínimo para la propiedad Value, es decir, el valor de esta propiedad cuando el cursor está en su parte mínima. (Recuerde que el cursor está en el mínimo, cuando está mas a la izquierda, caso del HScrollBar, o cuando está en la parte mas alta, caso del HScrollBar.
MouseIcon (*)
MousePointer (*)
Name (*)

SmallChange
Esta propiedad establece la variación de la propiedad Value cada vez que se hace click en las flechas superior o inferior de la barra de desplazamiento.
TabIndex (*)
TabStop (*)
Tag (*)
Top (*)

Value
Esta propiedad lee o establece el valor dado por la posición del cursor. Este valor tiene un mínimo, establecido por Min y un máximo, establecido por Max. Esta propiedad es la que se debe leer para conocer la posición del cursor.
Visible (*)
WhatsThisHelpID (*)
Width (*)

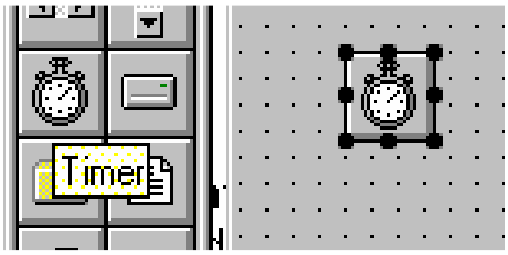
PROCEDIMIENTOS DE HScrollBar y VScrollBar
Change
DragDrop
DragOver
GotFocus
KeyDown
KeyPress
KeyUp
LostFocus
Scroll

Comentario El Procedimiento Change se produce cuando, tras mover el cursor, se suelta el botón del ratón. Esto produce el efecto de que el cambio que se tenga que producir con el movimiento del cursor no se realiza de una manera continua. El

procedimiento Scroll se realiza en el instante que se está moviendo el cursor. Por lo tanto, es este procedimiento el que se debe usar para conseguir el efecto de un cambio continuo mientras se está moviendo el cursor.

TIMER TEMPORIZADOR

Este objeto permite establecer temporizaciones. Presenta una novedad respecto a los controles estudiados hasta ahora. El control Timer solamente se ve durante el tiempo de diseño. En tiempo de ejecución, el control permanece invisible.
La temporización producida por el Timer es independiente de la velocidad de trabajo delordenador. (Casi independiente. El timer no es un reloj exacto, pero se le parece)
Se toma directamente de la caja de herramientas, y tiene el aspecto siguiente :



PROPIEDADES

Enabled (*)
Index (*)
Interval

El valor de esta propiedad nos dará el intervalo de tiempo (en milisegundos) en que se producirá un evento Timer y consecuentemente, realizará el procedimiento asociado a este evento. Si el valor de la propiedad Interval está a 0 (Predeterminado), no se produce el evento Timer. (El control Timer está deshabilitado cuando se pone la propiedad Interval = 0)

Left (*)
Name (*)
Tag (*)
Top (*)

PROCEDIMIENTOS

Timer
Se produce cada vez que se cumple un intervalo completo

Apunte enviado por correo: Procedimientos

Introducción a los procedimientos.

Con los procedimientos puede simplificar las tareas de programación, ya que divide los programas en componentes lógicos más pequeños que pueden convertirse en bloques básicos que le permiten mejorar y ampliar Visual Basic. Los procedimientos resultan muy útiles para condensar las tareas repetitivas o compartidas, como cálculos utilizados frecuentemente, manipulación de texto y controles, y operaciones con bases de datos. La base de una aplicación en Visual Basic la forman sus procedimientos conducidos por eventos. Hay dos ventajas principales cuando se programa con procedimientos:

- Los procedimientos le permiten dividir los programas en unidades lógicas discretas, cada una de las cuales se puede depurar más fácilmente que un programa entero sin procedimientos.
- Los procedimientos que se utilizan en un programa pueden actuar como bloques de construcción de otros programas, normalmente con pocas o ninguna modificación.

En Visual Basic se utilizan varios tipos de procedimientos, dos de ellos son:

- Procedimientos **Sub** que no devuelven un valor.
- Procedimientos **Function** que devuelven un valor.

Un procedimiento **Sub**, o **Function** contiene partes de código que se pueden ejecutar como una unidad.

Procedimientos Sub

Un procedimiento **Sub** es un bloque de código que se ejecuta como respuesta a un evento. Al dividir el código de un módulo en procedimientos **Sub**, es más sencillo encontrar o modificar el código de la aplicación. La sintaxis de un procedimiento **Sub** es la siguiente:

```
[Private|Public][Static]Sub nombreProcedimiento(parámetros)
      Instrucciones
      [Exit Sub]
      Instrucciones
End Sub
```

Cada vez que se llama al procedimiento se ejecutan las *instrucciones* que hay entre **Sub** y **End Sub**.
De forma predeterminada, los procedimientos **Sub** son **Public** en todos los módulos, lo que significa que se les puede llamar desde cualquier parte de la aplicación.
Los *parámetros* de un procedimiento son como las declaraciones de variables; se declaran valores que se pasan desde el procedimiento que hace la llamada.
Para hacer que un procedimiento (Sub o Function) sólo sea accesible desde los procedimientos del módulo al cual pertenece, hay que colocar al principio de la cabecera del procedimiento la palabra clave **Private**. Ejemplo:

```
Private Sub Proc_1(X AS Double, N AS Integer)
      ...
End Sub
```

Si no se especifica la palabra clave **Private** se supone que el procedimiento es **Public**, lo que significa que puede ser invocado desde otros módulos. Resulta muy útil en Visual Basic distinguir entre dos tipos de procedimientos **Sub**: *procedimientos generales* y *procedimientos de evento*.

Salir de un procedimiento Sub

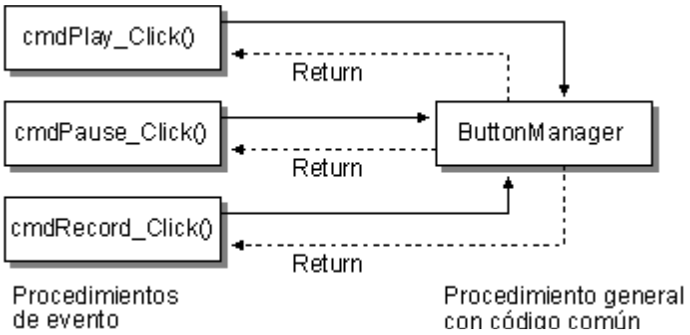
También puede salir de un procedimiento desde una estructura de control. **Exit Sub** puede aparecer tantas veces como sea necesario, en cualquier parte del cuerpo de un procedimiento **Sub**. **Exit Sub** es muy útil cuando el procedimiento ha realizado todo lo que tenía que hacer y se quiere volver inmediatamente.

Procedimientos generales

Un procedimiento general indica a la aplicación cómo realizar una tarea específica. Una vez que se define un procedimiento general, se le debe llamar específicamente desde la aplicación. Los procedimientos generales se crean porque, muchos procedimientos de evento distintos pueden necesitar que se lleven a cabo las mismas acciones. Es una buena estrategia de programación colocar las instrucciones comunes en un procedimiento distinto (un procedimiento general) y hacer que los procedimientos de evento lo llamen. Esto elimina la necesidad de duplicar código y también hace que la aplicación sea más fácil de mantener.

Ejemplo en la figura se muestra la utilización de un procedimiento general. El código de los eventos Click llama al procedimiento **Sub** ButtonManager que ejecuta su propio código y devuelve el control al procedimiento de evento Click.

Fig. Cómo llaman los procedimientos de evento a procedimientos generales



Procedimientos de evento

Un procedimiento de evento permanece inactivo hasta que se le llama para responder a eventos provocados por el usuario o desencadenados por el sistema. Cuando un objeto en Visual Basic reconoce que se ha producido un evento, llama automáticamente al procedimiento de evento utilizando el nombre correspondiente al evento. Como el nombre establece una asociación entre el objeto y el código, se dice que los procedimientos de evento están adjuntos a formularios y controles.

- Un procedimiento de evento de un control combina el nombre real del control (especificado en la propiedad **Name**), un carácter de subrayado (_) y el nombre del evento. Por ejemplo, si desea que un botón de comando llamado *cmdPlay* llame a un procedimiento de evento cuando se haga clic en él, utilice el procedimiento *cmdPlay_Click*.

Todos los procedimientos de evento utilizan la misma sintaxis general:

```
Private Sub NombreControl_NombreEvento(argumentos )
      Instrucciones
End Sub
```

Aunque puede escribir procedimientos de evento nuevos, es más sencillo usar los procedimientos de código que facilita Visual Basic, que incluyen automáticamente los nombres correctos de procedimiento. Puede seleccionar una plantilla en la ventana Editor de código si selecciona un objeto en el cuadro **Objeto** y selecciona un procedimiento en el cuadro **Procedimiento**.

- También es conveniente establecer la propiedad **Name** de los controles antes de empezar a escribir los procedimientos de evento para los mismos.

- Si cambias el nombre de un control tras vincularle un procedimiento, deberá cambiar también el nombre del procedimiento para que coincida con el nuevo nombre del control, de lo contrario, Visual Basic no será capaz de hacer coincidir el control con el procedimiento.
- Cuando el nombre de un procedimiento no coincide con el nombre de un control, se convierte en un procedimiento general.

Trabajar con procedimientos

Para crear un procedimiento general nuevo

Escribe el encabezado de un procedimiento en la ventana Código y presiona ENTRAR.
El encabezado del procedimiento puede ser tan simple como **Sub** o **Function** seguido de un nombre.
Por ejemplo, puede especificar cualquiera de los siguientes:

Sub ActualizarForm ()
Function ObtenerCoord ()

Visual Basic responde completando la plantilla del nuevo procedimiento.

Seleccionar procedimientos existentes

Para ver un procedimiento en el módulo actual:

- Selecciona **"(General)"** en el cuadro **Objeto** de la ventana Código y selecciona el procedimiento en el cuadro **Procedimiento**.
- bien para ver un procedimiento de evento, seleccione el objeto apropiado en el cuadro **Objeto** de la ventana Código y seleccione el evento en el cuadro **Procedimiento**.

Llamar a procedimientos Sub

- Un procedimiento **Sub** difiere de un procedimiento **Function** en que al procedimiento **Sub** no se le puede llamar mediante su nombre en una expresión.
- La llamada a un procedimiento **Sub** es una instrucción única. Además, un procedimiento **Sub** no devuelve un valor en su nombre como hace una función.
- Sin embargo, al igual que **Function**, un procedimiento **Sub** puede modificar los valores de las variables que se le pasan.
- Hay dos formas de llamar a un procedimiento **Sub**:

```
'Ambas instrucciones llaman a un Sub denominado MiProc.
Call MiProc (PrimerArgumento, SegundoArgumento)
MiProc PrimerArgumento, SegundoArgumento
```

- Observa que cuando utiliza la sintaxis **Call**, debe poner los argumentos entre paréntesis. **Si se omite la palabra clave** `Call`**, deberá también omitir los paréntesis alrededor de los argumentos.**

Llamar a procedimientos en otros módulos

- Se puede llamar a los procedimientos públicos de otros módulos desde cualquier parte del proyecto, necesitará especificar el módulo que contiene el procedimiento al que está llamando.
- Las técnicas para hacerlo varían, dependiendo de si el procedimiento está ubicado en un módulo estándar o de formulario.

Procedimientos en formularios

Todas las llamadas que se hacen desde fuera del módulo de formulario deben señalar al módulo de formulario que contiene el procedimiento.

Si un procedimiento llamado *SomeSub* está en un módulo de formulario llamado *Form1*, puede llamar al procedimiento en Form1 mediante esta instrucción:

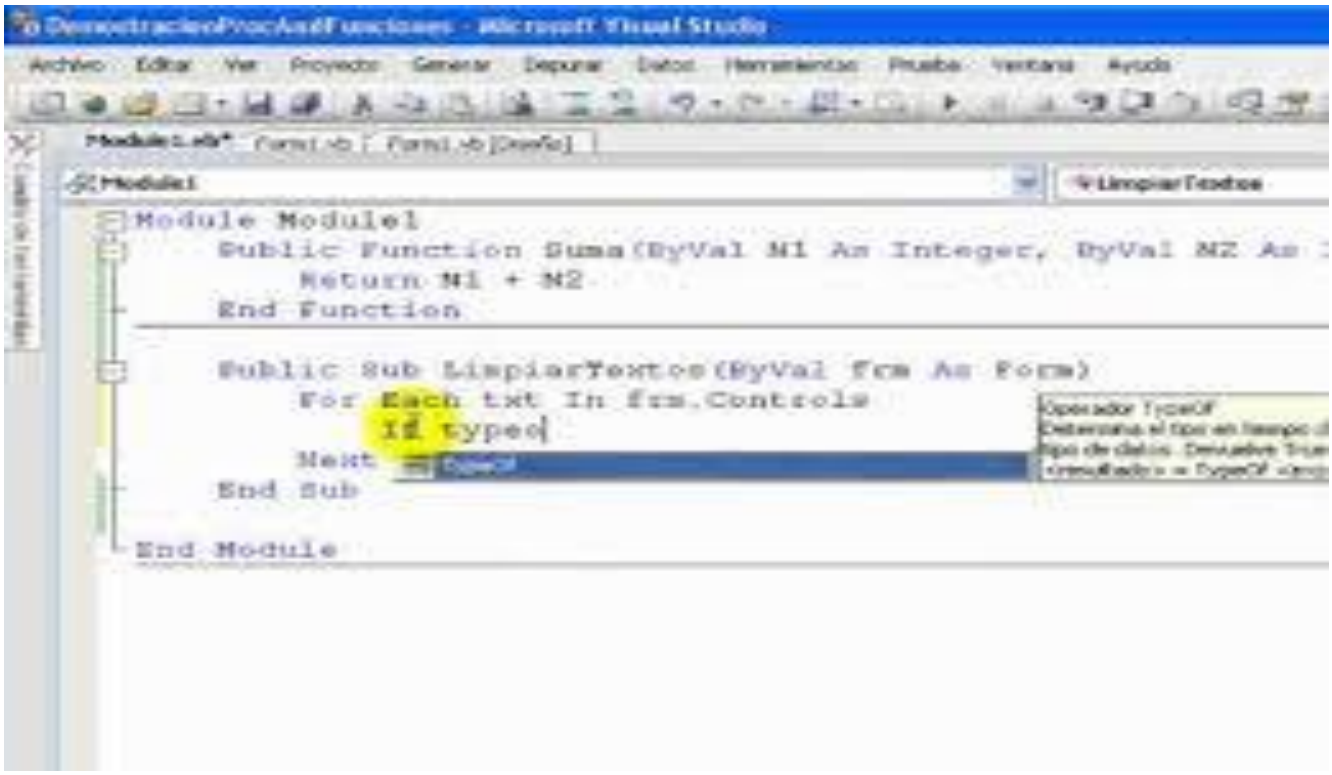
```
Call Form1.SomeSub(argumentos)
```

Procedimientos en módulos estándar

- *Si el nombre de un procedimiento es único, no necesita incluir el nombre del módulo en la llamada.*
- Una llamada desde dentro o fuera del módulo hará referencia a ese procedimiento único.
- Un procedimiento es único si sólo aparece en un lugar.
- *Si dos o más módulos contienen un procedimiento con el mismo nombre, debe calificarlo con el nombre del módulo.*
- Una llamada a un procedimiento común desde el mismo módulo ejecuta el procedimiento de ese módulo.
- Por ejemplo, con un procedimiento llamado *CommonName* en *Module1* y *Module2*, una llamada a *CommonName* desde *Module2* ejecutará el procedimiento *CommonName* de *Module2* y no el procedimiento *CommonName* de *Module1*.

- En una llamada a un nombre de procedimiento común desde otro módulo, debe especificar el módulo del procedimiento. Por ejemplo, si desea llamar al procedimiento *CommonName* de *Module2* desde *Module1*, utilice:

```
Module2.CommonName(argumentos)
```



Investigación: Guía del examen: definición de algunos controles, propiedades, métodos y eventos

Guia de Estudio para el Examen				
Controles	**Propiedades**	**Metodos**	**Eventos**	**Sentencias**
App	Alignment	LoadPicture()	Click()	Concatenación
ComboBox	BackColor	Path	Change()	Constantes de Color
Command Button	BorderStyle		Load()	Declaracion de procedimientos
frame	Caption		Scroll()	Flags
HScrollBar	Enabled		Timer()	If.. EndIf
Image	Font			If.. Then.. Else.. EndIf
Label	ForeColor			Llamadas a procedimientos
ListBox	Heigth			operadores aritméticos
Me	Interval			RGB()
Option Button	Left			Select.. Case
TextBox	ListIndex			Val()
Timer	Locked			Variables publicas
VScrollBar	Max			With.. End With
	MaxChange			
	Min			
	Name			
	PasswordChar			
	Picture			
	SmallChange			
	Stretch			
	TabIndex			
	Text			
	Top			
	Value			
	Visible			
	Width			

Controles

- *PictureBox*: Caja de imágenes
- *Label*: Etiqueta
- *TextBox*: Caja de texto
- *Frame*: Marco
- *CommandButton*: Botón de comando
- *CheckBox*: Casilla de verificación
- *OptionButton*: Botón de opción
- *ComboBox*: Lista desplegable
- *ListBox*: Lista
- *HScrollBar*: Barra de desplazamiento horizontal
- *VScrollBar*: Barra de desplazamiento vertical

- *Timer*: Temporizador
- *DriveListBox*: Lista de unidades de disco
- *DirListBox*: Lista de directorios
- *FileListBox*: Lista de archivos
- *Shape*: Figura
- *Line*: Línea
- *Image*: Imagen
- *Data*: Conexión a origen de datos

Propiedades

TEXTBOX:

Text: texto que aparecerá en el control.

Name: nombre del control.

Multiline: nos permite introducir varias lineas de texto.

Alignment: Alineación que tendrá el texto dentro del control que puede ser izquierdo, derecho, centrado.

Visible: si esta propiedad esta en falso la caja de texto no sera visible cuando este en ejecución el programa. si está en verdadero si se podrá ver.

Maxlength: numero máximo de caracteres que tendrá el control.

Looked: Con esta propiedad podemos bloquear el control para que el usuario no pueda escribir ni modificar.

Backcolor: Color que tendrá el fondo de la caja de texto.

Forecolor: Es el color de la letra que tendrá el control.

Font: tipo y tamaño que contendrá el control.

Height,Left,Top,Width : Se refieren al tamaño del Espacio reservado para las Text Box.

LABEL:

Caption: texto que contendrá el control.

BorderStyle: borde al rededor del texto.

BackStyle: borde transparente o no transparente.

Backcolor: Para cambiar color del fondo.

Visible : Si está en True el control está visible si está en False está oculto.

Name: es el nombre del control y sirve para hacer referencia al mismo en el código, como todos los controles.

Enabled: Si está en True (Verdadero) el control está activado y se puede utilizar normalmente, si se encuentra en False, el control está desactivado.

FontName: El nombre de la fuente que utilizará el texto del control. Podemos elegir las que tengamos instaladas en el sistema.

Forecolor: indica el color del Texto.

Height y Width: Ancho y alto del Label.

ToolTipText: muestra el mensaje de descripción cuando pasamos el mouse por encima del control.

COMMANDBUTTON:

Caption: texto para el usuario.

Enable: Inhavilita o habilita el control con el fin de que ese disponible para el usuario.

Style: cuando esta en 1 habilita al backcolor y cuando esta en 0 lo deshabilita.

PICTURE E IMAGE:

Name: Especifíca el nombre del control para poder referenciarlo e identificarlo.

Appearance: Esta propiedad determina si el Image posee o no efecto 3d con respecto a su apariencia. Los valores son: 1 - 3D y 0 - None. Para que esta propiedad se pueda utilizar, la propiedad BorderStyle debe estar con el valor 1.

BorderStyle: Determina si el control Image posee o no un borde. Al igual que el anterior tiene dos posibles valores, 0 sin borde o 1 con borde.

Picture: Esta es la propiedad principal del control, que también es la propiedad por defecto o default. Picture es la que establece la imagen o gráfico que mostrará el control.

Stretch: Esta es una de las propiedades mas importantes. Si está en True la imagen se ajustará al tamaño que posea el control Image, si está en False el control Image es el que se adaptará al tamaño y dimensiones de la imagen.

Las demás propiedades son las clásicas y comunes para la mayoría de los controles, como la propiedad Index, Visible, Enabled, ToolTipText, Width, Height, etc...

FRAME:

Name : Este es el nombre como se le reconocera al objeto durante el programa, se acostumbra escribir frm antes del nombre para saber que es un Frame (Ej: frmFondo)..

Caption : Este es el mensaje que se quiere que aparesca en el Frame.

Height,Left,Top,Width : Se refieren al tamaño del Espacio reservado para los Frames.

Font : Permite escoger el tipo de letra, tamaño y estilo de la letras a usar.

BorderStyle: si esta en 0 no dibuja el recuadro.

CHECKBOX Y OPTIONBUTTON :

Name : Este es el nombre como se le reconocera al objeto durante el programa, se acostumbra escribir chk o rdb antes del nombre para saber que es un check box o radio Button (Ej: chkop1 o rdbop1).

Enable : Este parametro permite que el Boton este habilitado o deshabilitado.

Value : Indica si esta precionado o no el boton.

Height,Left,Top,Width : Se refieren al tamaño del Espacio reservado para los Check box y los option Button.

Caption : Este es el mensaje que se quiere que aparesca a la par del radio Button o del Check Box.

LITSBOX:

Name : Este es el nombre como se le reconocera al objeto durante el programa, se acostumbra escribir lst antes del nombre para saber que es un List Box (Ej: lstLista).

List : Aqui podemos ingresar los elementos nuevos al List Box.

Font : Permite escoger el tipo de letra, tamaño y estilo de la letras a usar.

Sorted: si el valor es verdadero la lista aparecera en orden alfabetico.

Height,Left,Top,Width : Se refieren al tamaño del Espacio reservado para los List Box.

SCROLL BAR:

Name : Este es el nombre como se le reconocera al objeto durante el programa, se acostumbra escribir scb antes del nombre para saber que es un Scroll Bar (Ej: scbFila).

Max : Este indica el valor maximo que puede alcanzar el Scroll al moverse.

Min : Es el valor minim con el cual empieza la barra de Scroll.

Height,Left,Top,Width : Se refieren al tamaño del Espacio reservado para los Scroll Bar.

SHAPE:

Shape: esta propiedad es la forma que tomara nuestro control.

Bordercolor: color del borde.

Borderstyle: estilo del borde.

Border whidth: ancho del borde.

Fillcolor: este es el fondo, esta funciona si filestyle en opaco.

Eventos y Metodos

- *Click*: ocurre cuando se presiona y suelta un botón del mouse sobre un objeto.
- *DblClick*: ocurre cuando se presiona y suelta dos veces un botón del mouse sobre un objeto.
- *DragDrop*: ocurre al arrastrar y soltar un determinado objeto con el mouse.
- *DragOver*: ocurre si una operación de arrastrar y soltar está en curso.
- *GotFocus*: ocurre cuando un objeto recibe el control o foco, ya sea mediante una acción del usuario como hacer clic en un objeto ventana, o cambiando el foco de objeto desde el programa, mediante el método SetFocus.
- *LostFocus*: contrario al anterior, este evento ocurre cuando el objeto pierde el enfoque, sea mediante acción del usuario o efectuado desde la aplicación.
- *KeyDown*: ocurre cuando el usuario mantiene presionada una tecla.
- *KeyUp*: ocurre cuando el usuario deja de presionar una tecla. Este evento sucede precisamente al terminar el evento KeyDown.
- *KeyPress*: ocurre como cuando se presiona y suelta una tecla.
- *MouseDown*: ocurre cuando el usuario presiona un botón del mouse.
- *MouseUp*: se produce cuando el usuario suelta el botón del mouse.
- *MouseMove*: este evento ocurre mientras el usuario mueve o desplaza el puntero del mouse sobre un objeto.

ANEXO DE APLICACIONES

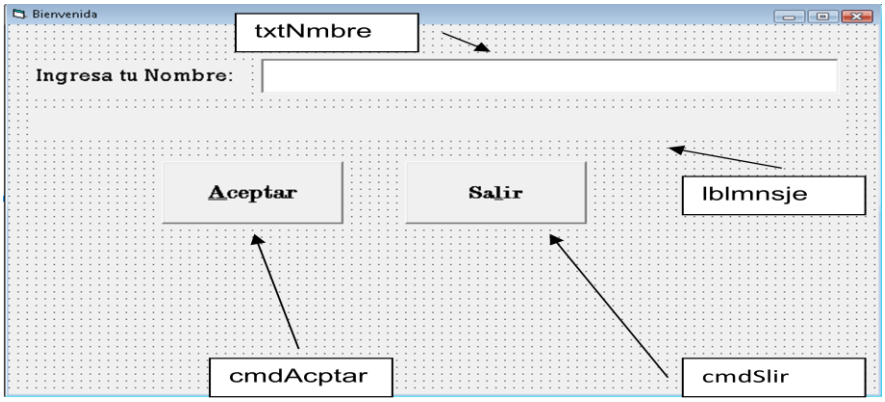
Aplicación 1: Bienvenida.

El programa debe recibir de entrada el nombre del usuario y dar un mensaje de bienvenida personalizándolo con el nombre recibido. Es una versión mejorada del tradicional "Hola Mundo".

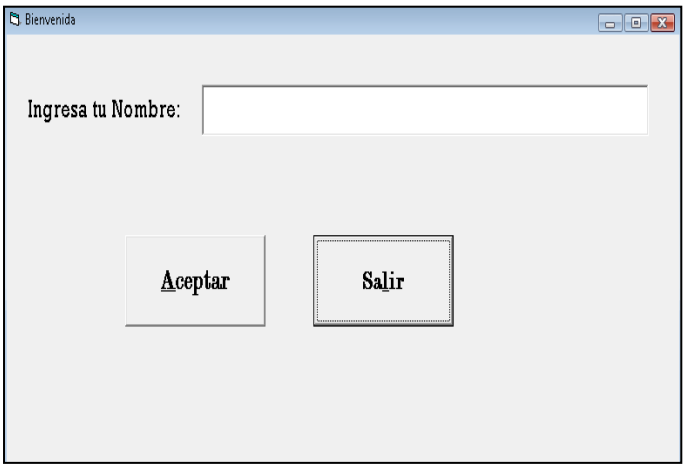
Temas.

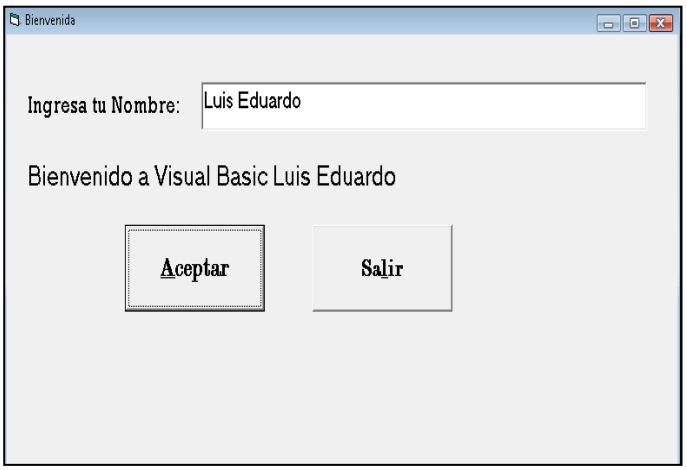
Controles: *Command Button, Label, TextBox*
Propiedades: *Caption, Font, Name, Text*
Eventos: *Click()*
Sentencias o Instrucciones: *Concatenación*

Diseño



Pantallas en ejecución.





Código

```
Private Sub cmdAcptar_Click()

    lblmnsje.Caption = "Bienvenido a Visual Basic " & txtNmbre.Text

End Sub

Private Sub cmdSlir_Click()

    End

End Sub
```

****

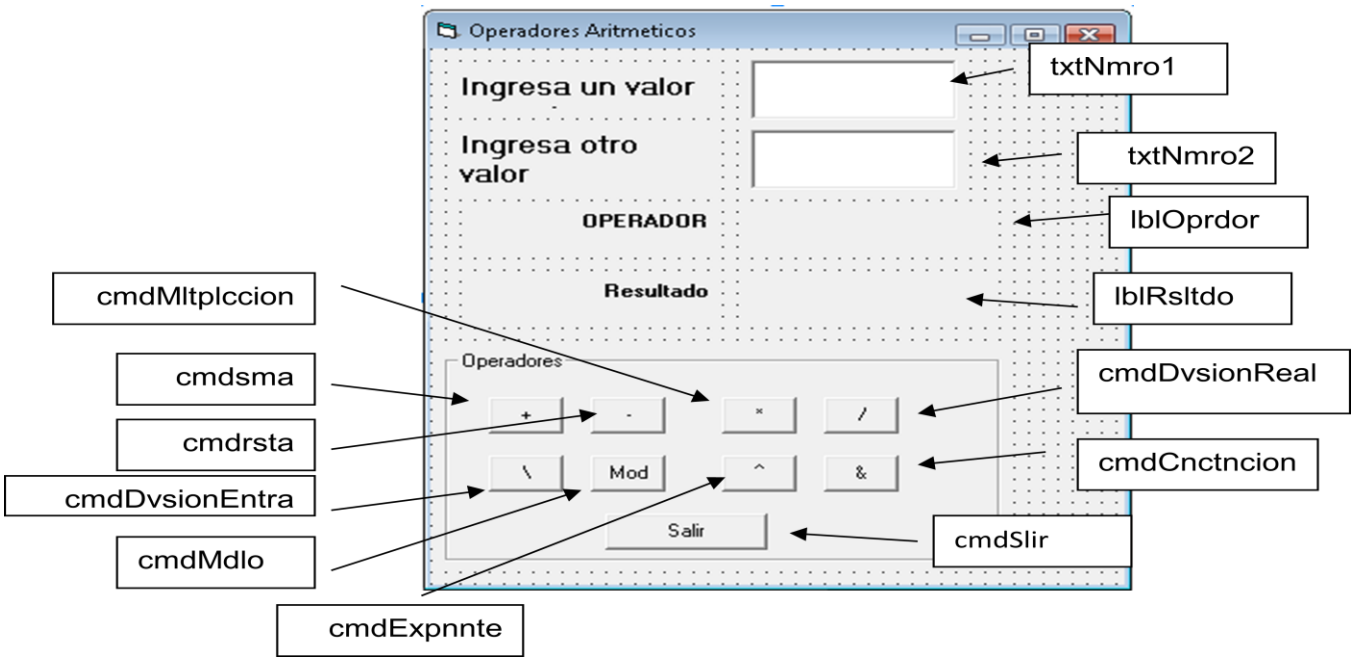
Aplicación 2: Operadores Aritmeticos

La aplicación recibirá de entrada dos valores numéricos y el usuario tendrá la opción de seleccionar la operación matemática a realizar con ellos, a través de botones de comando, donde con cada uno de ellos podrá hacer: suma, resta, multiplicación, divisón real, división entera, divisón en módulo, exponenciación y concatenación.

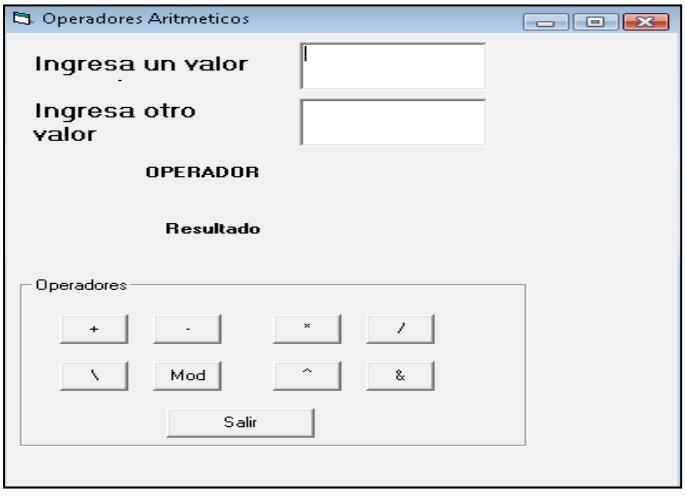
Temas.

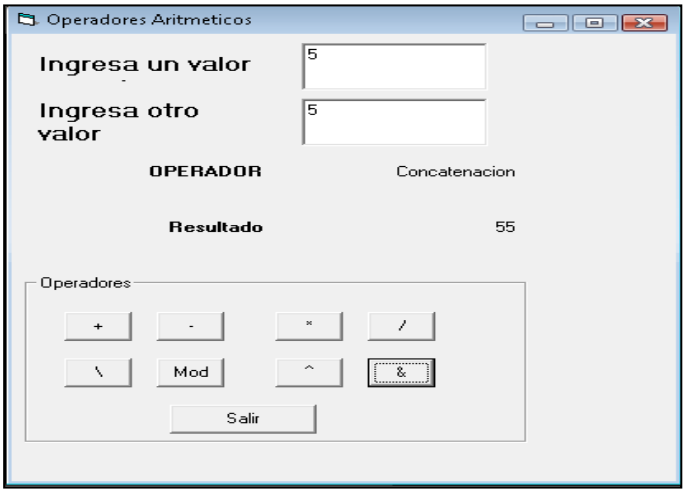
Controles: *Command Button, Label, TextBox*
Propiedades: *Alignment, BackColor, BorderStyle, Caption, Font, Name, Text*
Eventos: *Click()*
Sentencias o Instrucciones: *Concatenación, operadores aritméticos, Función Val()*

Diseño



Pantallas en ejecución.





Código

```
Private Sub cmdCnctncion_Click()
lblOprdor.Caption = "Concatenacion"
lblRsltdo.Caption = Val(txtNmro1.Text) & Val(txtNmro2.Text)
End Sub

Private Sub cmdDvsionEntra_Click()
lblOprdor.Caption = "Division Entera"
lblRsltdo.Caption = Val(txtNmro1.Text) \ Val(txtNmro2.Text)
End Sub

Private Sub cmdDvsionReal_Click()
lblOprdor.Caption = "Division Real"
lblRsltdo.Caption = Val(txtNmro1.Text) / Val(txtNmro2.Text)
End Sub
Private Sub cmdExpnnte_Click()
lblOprdor.Caption = "Exponente"
lblRsltdo.Caption = Val(txtNmro1.Text) ^ Val(txtNmro2.Text)
End Sub

Private Sub cmdMdlo_Click()
lblOprdor.Caption = "Modulo"
lblRsltdo.Caption = Val(txtNmro1.Text) Mod Val(txtNmro2.Text)
End Sub

Private Sub cmdMltplccion_Click()
lblOprdor.Caption = "Multiplicaciòn"
lblRsltdo.Caption = Val(txtNmro1.Text) * Val(txtNmro2.Text)
End Sub

Private Sub cmdrsta_Click()
lblOprdor.Caption = "Resta"
lblRsltdo.Caption = Val(txtNmro1.Text) - Val(txtNmro2.Text)
End Sub

Private Sub cmdSlir_Click()
End
End Sub

Private Sub cmdsma_Click()
lblOprdor.Caption = "Suma"
lblRsltdo.Caption = Val(txtNmro1.Text) + Val(txtNmro2.Text)
End Sub
```

****

Aplicación 3: Figuras Geometricas

La aplicación debe permitir al usuario seleccionar el nombre de una figura geométrica, introducir los datos necesarios y calcular el área de dicha figura. Las figuras a manejar son: rectángulo, círculo, triángulo, rombo, trapecio, polígono regular (hexágono).

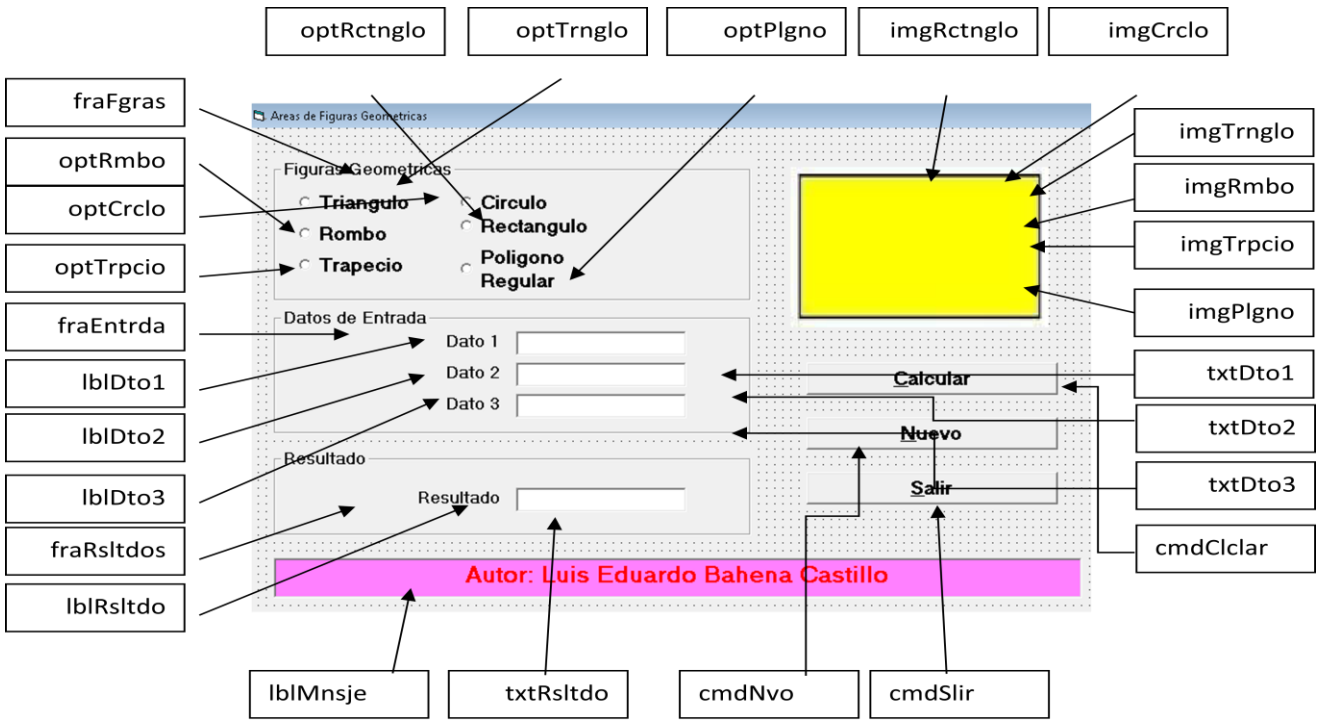
Características.

- Se deben utilizar botones de opción (*Option Button*) para seleccionar el nombre.
- Agrupar los botones de comando en un contenedor *frame.*
- Agrupar en otro *frame* los correspondientes *Label* y *TextBox* para entrada de datos. Al iniciar el programa, este *frame* debe estar invisible.
- Agrupar en un tercer *frame* los *Label* y *TextBox* para salida de datos. Al iniciar el programa, este *frame* debe estar invisible.
- Debe haber tres botones de comando (*Command Button*): *Calcular, Nuevo y Salir*, de los cuales solo *Calcular* al iniciar la aplicación debe estar inhabilitado.
- El orden del foco debe ser: 0 – Botón Nuevo, 1 – Botón Salir, 2 – Botón Calcular y en seguida los botones de opción.
- Una vez seleccionada la figura en los botones de opción hacer lo siguiente:
  - ✓ Debe aparecer una imagen correspondiente a la figura geométrica, utilizar el control *image*.
  - ✓ Hacer visible el *frame* de entrada de datos para permitir la captura de datos, y que solicite los datos de entrada pertinentes a la figura que haya sido seleccionada (por ejemplo: base y altura, si se seleccionó Triángulo)
  - ✓ El botón de comando *Calcular*, debe habilitarse para poder ser utilizado.
- Al hacer clic en *Calcular*, debe hacer visible el *frame* de salida de datos y mostrar el área calculada.
- El botón de comando *Nuevo* es para permitir hacer otro cálculo, el cual borrará los datos, dejando el formulario con el mismo aspecto de cuando arranca o se carga la aplicación.
- El botón de comando Salir, es para terminar la ejecución de la aplicación.
- En la parte inferior del formulario debe haber una etiqueta donde indicará el nombre del autor.
- La barra de título debe llevar el nombre de la aplicación.

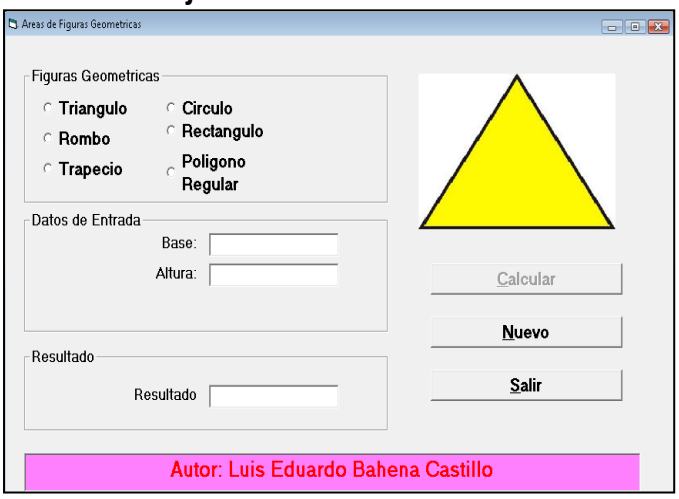
Temas.

Controles: *Command Button, frame, Image, Label, TextBox, Option Button.*
Propiedades: *Alignment, BackColor, BorderStyle, Caption, Enabled, Font, ForeColor, Locked, Name, Picture, Stretch, TabIndex, Text, Value, Visible*
Sentencias: *Select Case….End Select, Variables públicas*

Diseño



Pantallas en ejecución.



Código

```
Public FiguraSeleccionada As Integer

Public Sub ocultaImagenes()
    'Oculta las Imagenes'
    imgTrnglo.Visible = False
    imgRmbo.Visible = False
    imgTrpcio.Visible = False
    imgCrclo.Visible = False
    imgRctnglo.Visible = False
    imgPlgno.Visible = False
End Sub
Private Sub cmdClclar_Click()
    Select Case FiguraSeleccionada
        Case 0   'Triangulo
            txtRsltdo.Text = txtDto1.Text * txtDto2.Text / 2
        Case 1   'Rombo
            txtRsltdo.Text = txtDto1.Text * txtDto2.Text / 2
        Case 2   'Trapecio
            txtRsltdo.Text = txtDto1.Text + txtDto2.Text * txtDto3.Text / 2
        Case 3   'Circulo
            txtRsltdo.Text = 3.1416 * txtDto1.Text ^ 2
        Case 4   'Rectangulo
            txtRsltdo.Text = txtDto1.Text * txtDto2.Text
        Case 5   'Poligono
            txtRsltdo.Text = txtDto1.Text * txtDto2.Text / 2
    End Select
End Sub

Private Sub cmdNvo_Click()
    'Hacer visible el frame de opciones'
     fraFgras.Enabled = True

    'Habilita boton Calcular
     cmdClclar.Enabled = True

    'Ocultar frame de datos de entrada
     fraEntrda.Visible = False

    'Ocultar el frame de resultado'
```

```
     fraRsltdo.Visible = False

     'Hace la llamada a la subrutina para ocultar imagenes
     ocultaImagenes

     'Limpiar datos y resultados
     txtDto1.Text = ""
     txtDto2.Text = ""
     txtDto3.Text = ""
     txtRsltdo.Text = ""

End Sub

Private Sub Form_Load()
    'Oculta los frames'
    fraEntrda.Visible = False
    fraRsltdo.Visible = False

    'Inhabilita boton calcular'
    cmdClclar.Enabled = False

    'Hace la llamada a la subrutina para ocultar imagenes
    ocultaImagenes

    'Deselecciona los botones de opcion'
    optTrnglo.Value = False
    optRmbo.Value = False
    optTrpcio.Value = False
    optCrclo.Value = False
    optRctnglo.Value = False
    optPlgno.Value = False

    'Hacer visible el frame de entrada'
     fraEntrda.Visible = True
     fraRsltdo.Visible = True
     imgTrnglo.Visible = True

    'Cambiar leyendas de datos
    lblDto1.Caption = "Base:"
    lblDto2.Caption = "Altura:"

    'Ocultar label y text que no se ocupan
    lblDto3.Visible = False
    txtDto3.Visible = False

    'Asigna valor a la variable publica
    FiguraSeleccionada = 0

End Sub
Private Sub optCrclo_Click()
     'Hacer visible el frame de entrada'
     fraEntrda.Visible = True
     fraRsltdo.Visible = True
     imgCrclo.Visible = True

    'Cambiar leyendas de datos
    lblDto1.Caption = "Radio:"

    'Ocultar label y text que no se ocupan
    lblDto2.Caption = False
    txtDto2.Visible = False
```

```
    lblDto3.Caption = False
    txtDto3.Visible = False

    'Asigna valor a la variable publica
    FiguraSeleccionada = 3
End Sub

Private Sub optPlgno_Click()
     'Hacer visible el frame de entrada'
     fraEntrda.Visible = True
     fraRsltdo.Visible = True
     imgPlgno.Visible = True

    'Cambiar leyendas de datos
    lblDto1.Caption = "Perimetro:"
    lblDto2.Caption = "Apotema:"

    'Ocultar label y text que no se ocupan
    lblDto3.Visible = False
    txtDto3.Visible = False

    'Asigna valor a la variable publica
    FiguraSeleccionada = 5
End Sub

Private Sub optRctnglo_Click()
     'Hacer visible el frame de entrada'
     fraEntrda.Visible = True
     fraRsltdo.Visible = True
     imgRctnglo.Visible = True

    'Cambiar leyendas de datos
    lblDto1.Caption = "Base:"
    lblDto2.Caption = "Altura:"

    'Ocultar label y text que no se ocupan
    lblDto3.Visible = False
    txtDto3.Visible = False

    'Asigna valor a la variable publica
    FiguraSeleccionada = 4
End Sub

Private Sub optRmbo_Click()
     'Hacer visible el frame de entrada'
     fraEntrda.Visible = True
     fraRsltdo.Visible = True
     imgRmbo.Visible = True

    'Cambiar leyendas de datos
    lblDto1.Caption = "Diagonal Mayor:"
    lblDto2.Caption = "Diagonal Menor:"

    'Ocultar label y text que no se ocupan
    lblDto3.Visible = False
    txtDto3.Visible = False

    'Asigna valor a la variable publica
    FiguraSeleccionada = 1

End Sub
```

```
Private Sub optTrnglo_Click()
     'Hacer visible el frame de entrada'
     fraEntrda.Visible = True
     fraRsltdo.Visible = True
     imgTrnglo.Visible = True

    'Cambiar leyendas de datos
    lblDto1.Caption = "Base:"
    lblDto2.Caption = "Altura:"

    'Ocultar label y text que no se ocupan
    lblDto3.Visible = False
    txtDto3.Visible = False

    'Asigna valor a la variable publica
    FiguraSeleccionada = 0

End Sub

Private Sub optTrpcio_Click()
     'Hacer visible el frame de entrada'
     fraEntrda.Visible = True
     fraRsltdo.Visible = True
     imgTrpcio.Visible = True

    'Cambiar leyendas de datos
    lblDto1.Caption = "Base Mayor:"
    lblDto2.Caption = "Base Menor:"
    lblDto3.Caption = "Altura:"

    'Hacer visible el label y text 3
    lblDto3.Visible = True
    txtDto3.Visible = True

    'Asigna valor a la variable publica
    FiguraSeleccionada = 2
End Sub
Private Sub cmdSlir_Click()
    End
End Sub
```

****

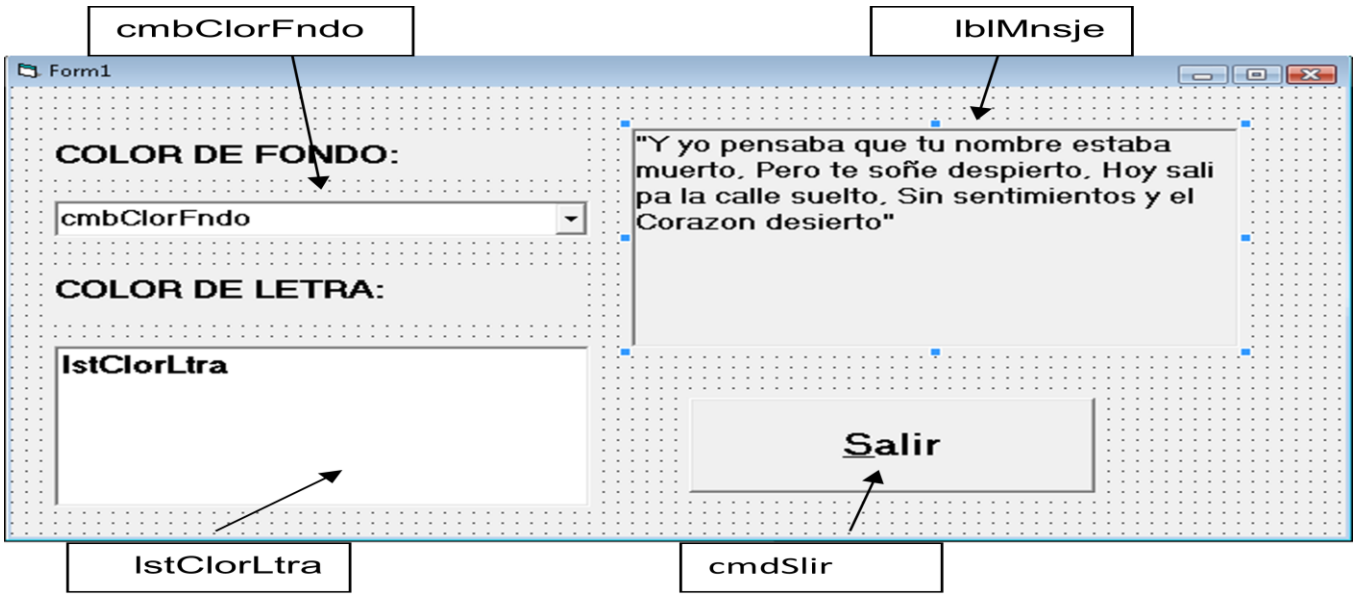
Aplicación No. 4. Demo Listas y Colores

La aplicación demostrará el uso de Listas y aplicación de colores utilizando el sistema RGB. Se utilizarán los controles *ComboBox* y *ListBox*, en el primero se desplegarán una lista de colores para seleccionar el color a aplicar al fondo de una etiqueta, y en la ListBox se seleccionará el color de primer plano (color de letra) para la misma etiqueta.

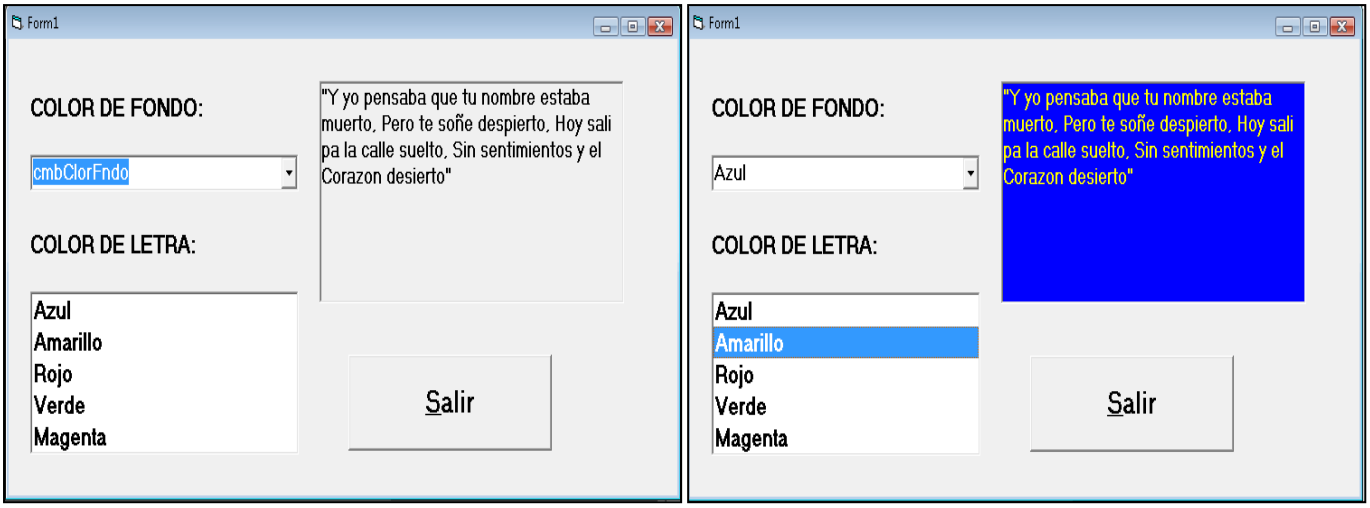
Temas:

Controles: *ListBox, ComboBox, Label*
Propiedades: *BackColor, ForeColor, ListIndex*
Eventos: *Click(), Load()*
Métodos: *.AddItem()*
Sentencias: *Función RGB(), constantes de color de VB, Select Case, With..End With*

Diseño



Pantallas en ejecución.



Código

```
Private Sub Form_Load()
    'Carga colores de fondo
    cmbClorFndo.AddItem "Azul"
    cmbClorFndo.AddItem "Amarillo"
    cmbClorFndo.AddItem "Rojo"
    cmbClorFndo.AddItem "Verde"
    cmbClorFndo.AddItem "Magenta"

    'Carga colores de letra
    With lstClorLtra
    .AddItem "Azul"
    .AddItem "Amarillo"
    .AddItem "Rojo"
    .AddItem "Verde"
    .AddItem "Magenta"
    End With
End Sub

Private Sub cmbClorFndo_Click()
    Select Case cmbClorFndo.ListIndex
        Case 0 'azul
    lblMnsje.BackColor = vbBlue
        Case 1 'amarillo
    lblMnsje.BackColor = vbYellow
        Case 2 'rojo
    lblMnsje.BackColor = vbRed
        Case 3 'verde
    lblMnsje.BackColor = vbGreen
        Case 4 'magenta
    lblMnsje.BackColor = RGB(255, 0, 255)
    End Select
End Sub
Private Sub lstClorLtra_Click()
    Select Case lstClorLtra.ListIndex
        Case 0 'azul
    lblMnsje.ForeColor = vbBlue
        Case 1 'amarillo
    lblMnsje.ForeColor = vbYellow
        Case 2 'rojo
    lblMnsje.ForeColor = vbRed
        Case 3 'verde
    lblMnsje.ForeColor = vbGreen
        Case 4 'magenta
    lblMnsje.ForeColor = RGB(255, 0, 255)
    End Select
End Sub
Private Sub cmdSlir_Click()
    End
End Sub
```

****

Aplicación No. 5. Demo Scroll y RGB

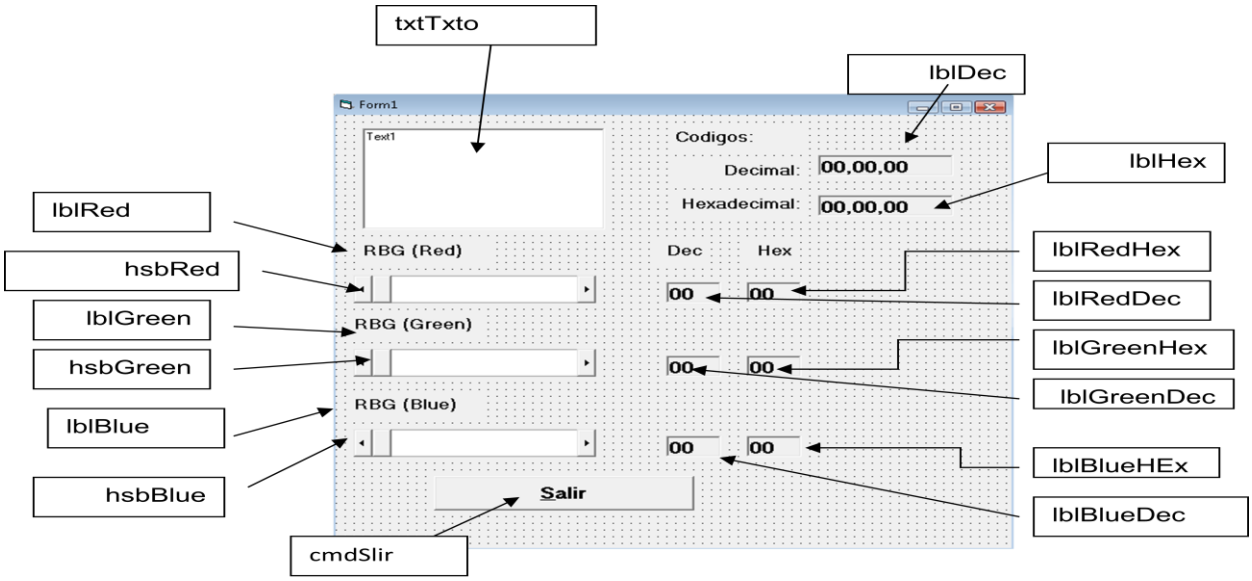
La aplicación demostrará el uso de una barra de desplazamiento horizontal, y el uso de los colores en Visual Basic con el sistema RGB (Red – Green – Blue). Constará de tres barras, cada una para cada color (rojo, verde y azul), con las cuales el usuario al desplazarse se irá asignando el color respectivo en el fondo de una caja de texto. Nos mostrará el código RGB correspondiente así como el código en Hexadecimal.

Temas:

Controles: HScrollBar, Label, TextBox, CommandButton

Propiedades: BackColor, ForeColor, Caption, Text, Value, Max, Min, MaxChange, SmallChange
Eventos: Click(), Load(), Change(), Scroll()
Sentencias: RGB(), constantes de color de VB, Variables públicas, declaración de procedimientos Sub públicos, llamadas a procedimientos Sub públicos, códigos de color en hexagecimal, concatenación

Diseño



Pantallas en ejecución.

Código

```
Public Red, Green, Blue As Integer
Private Sub InicializarScroll(hsb As HScrollBar)
hsb.Max = 255     'valor max
```

```
hsb.Min = 0       'valor min
hsb.SmallChange = 1  'valor que salta al click
hsb.LargeChange = 10 'valor que salgo cuando se mueve
End Sub

Private Sub ActualizarEtiquetas(lbl As Label)
  lbl.BackColor = &H0  'Fondo Negro
  lbl.ForeColor = &HFFFF& 'Letra Amarillo 255,255,0
End Sub
Private Sub ActualizarColorTexto()
txtTxto.BackColor = RGB(Red, Green, Blue)
lblDec.Caption = Red & "-" & Green & "-" & Blue
lblHex.Caption = "H" & Hex(Blue) & "-" & Hex(Green) & "-" & Hex(Blue)
End Sub
Private Sub ActualizarBlue()
Blue = hsbBlue.Value
lblBlueDec.Caption = Blue
lblBlueDec.BackColor = RGB(Blue, 0, 0)
lblBlueHEx.Caption = Hex(Blue)
lblBlueHEx.BackColor = RGB(Blue, 0, 0)
ActualizarColorTexto
End Sub
Private Sub ActualizarRed()
Red = hsbRed.Value
lblRedDec.Caption = Red
lblRedDec.BackColor = RGB(Red, 0, 0)
lblRedHex.Caption = Hex(Red)
lblRedHex.BackColor = RGB(Red, 0, 0)
ActualizarColorTexto
End Sub
Private Sub ActualizarGreen()
Green = hsbGreen.Value
lblGreenDec.Caption = Green
lblGreenDec.BackColor = RGB(Green, 0, 0)
lblGreenHex.Caption = Hex(Green)
lblGreenHex.BackColor = RGB(Green, 0, 0)
ActualizarColorTexto
End Sub

Private Sub Form_Load()
txtTxto.Text = "El BROMAS EN CLASE CON EL 3BPM"

'Establece color de fondo de las etiquetas
lblRed.BackColor = &HFF&        'Fondo Rojo 255,0,0
lblGreen.BackColor = &HFF00&    'Fondo Verde 0,255,0
lblBlue.BackColor = &HFF0000    'Fondo Azul 0,0,255

'Llmada al procedimiento con parametros
ActualizarEtiquetas lblRedDec
ActualizarEtiquetas lblRedHex
ActualizarEtiquetas lblBlueDec
ActualizarEtiquetas lblBlueHEx
ActualizarEtiquetas lblGreenDec
ActualizarEtiquetas lblGreenHex

InicializarScroll hsbRed
InicializarScroll hsbGreen
InicializarScroll hsbBlue
'Inicia variables publicas
Red = 0
Blue = 0
```

```
Green = 0
End Sub

Private Sub hsbBlue_Change()
Blue = hsbBlue.Value
lblBlueDec.Caption = Blue
lblBlueDec.BackColor = RGB(Blue, 0, 0)
lblBlueHEx.Caption = Hex(Blue)
lblBlueHEx.BackColor = RGB(Blue, 0, 0)
ActualizarColorTexto
End Sub
Private Sub hsbBlue_Scroll()
Blue = hsbRed.Value
lblBlueDec.Caption = Red
lblBlueDec.BackColor = RGB(Red, 0, 0)
lblBlueHEx.Caption = Hex(Red)
lblBlueHEx.BackColor = RGB(Red, 0, 0)
ActualizarColorTexto
End Sub

Private Sub hsbGreen_Change()
Green = hsbGreen.Value
lblGreenDec.Caption = Green
lblGreenDec.BackColor = RGB(Green, 0, 0)
lblGreenHex.Caption = Hex(Green)
lblGreenHex.BackColor = RGB(Green, 0, 0)
ActualizarColorTexto
End Sub
Private Sub hsbGreen_Scroll()
Green = hsbGreen.Value
lblGreenDec.Caption = Green
lblGreenDec.BackColor = RGB(Green, 0, 0)
lblGreenHex.Caption = Hex(Green)
lblGreenHex.BackColor = RGB(Green, 0, 0)
ActualizarColorTexto
End Sub

Private Sub hsbRed_Change()
Red = hsbRed.Value
lblRedDec.Caption = Red
lblRedDec.BackColor = RGB(Red, 0, 0)
lblRedHex.Caption = Hex(Red)
lblRedHex.BackColor = RGB(Red, 0, 0)
ActualizarColorTexto
End Sub
Private Sub hsbRed_Scroll()
Red = hsbRed.Value
lblRedDec.Caption = Red
lblRedDec.BackColor = RGB(Red, 0, 0)
lblRedHex.Caption = Hex(Red)
lblRedHex.BackColor = RGB(Red, 0, 0)
ActualizarColorTexto
End Sub

Private Sub cmdSlir_Click()
    End
End Sub
```

****

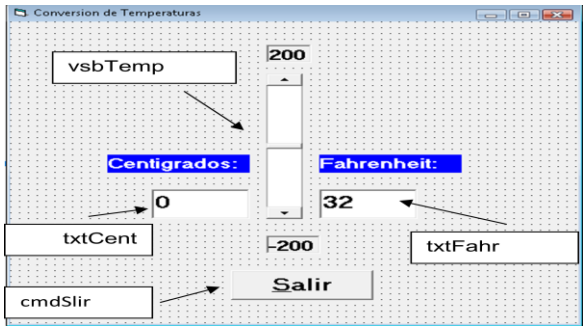
Aplicación No. 6. Temperaturas Scroll

Utilizando una barra de desplazamiento vertical, el usuario podrá desplazarse en ella y simultáneamente aparecerá en un Label la temperatura en grados centígrados de acuerdo al valor de la barra, y al mismo tiempo hará la conversión a su equivalente en grados Fahrenheit, la cual aparecerá en otra etiqueta Label.

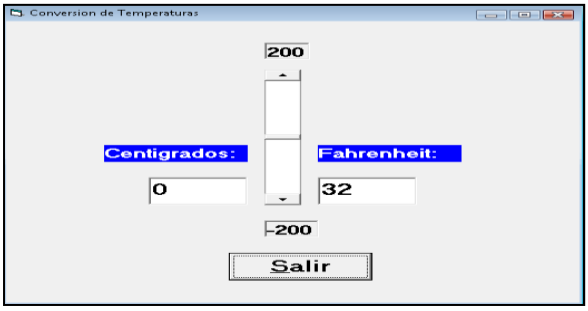
Temas:

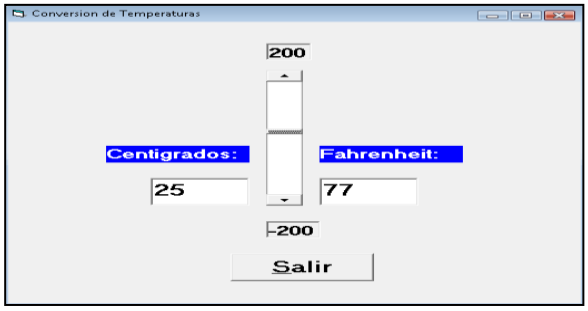
Controles: VScrollBar, Label.
Propiedades: BackColor, ForeColor, Caption, Value, Max, Min, MaxChange, SmallChange
Eventos: Load(), Change(), Scroll()
Sentencias: Variables públicas, declaración de procedimientos Sub públicos, llamadas a procedimientos Sub públicos, operadores aritméticos.

Diseño



Pantallas en ejecución.





Código

```
Private Sub cmdSlir_Click()
     End
End Sub
'Calculate the temperature
Private Sub vsbTemp_Change()
  txtCent.Text = vsbTemp.Value
  txtFahr.Text = 32 + 1.8 * vsbTemp.Value
End Sub
Private Sub Form_Load()
  Form1.Top = (Screen.Height - Form1.Height) / 2
  Form1.Left = (Screen.Width - Form1.Width) / 2
End Sub
Private Sub InicializarScroll(hsb As VScrollBar)
vsb.Max = 200     'valor max
vsb.Min = -200       'valor min
vsb.SmallChange = 1  'valor que salta al click
vsb.LargeChange = 10 'valor que salgo cuando se mueve
End Sub
```

****

Aplicación 7. Demo Timer

Demostrará el uso del control Timer. El usuario dispondrá de un botón de comando con el cual activará el efecto de parpadeo (video inverso) en un objeto Label, (se alternarán los colores de fondo y de letra en un periodo de tiempo de un segundo), asimismo el usuario podrá detener el efecto a través de otro botón de comando.
Utilizando otro control Timer, al activar el efecto de parpadeo (video inverso), la etiqueta se posicionará en la parte más baja del formulario y se desplazará de abajo hacia arriba, dicho movimiento lo realizará en periodos de tiempo de 100 milisegundos; en cuanto desaparezca la etiqueta del formulario, debe aparecer de nuevo en la parte más baja del formulario y otra vez desplazarse hacia arriba.
Con otro botón de comando se podrá detener el intercambio de colores y el desplazamiento del control Label.

Temas:

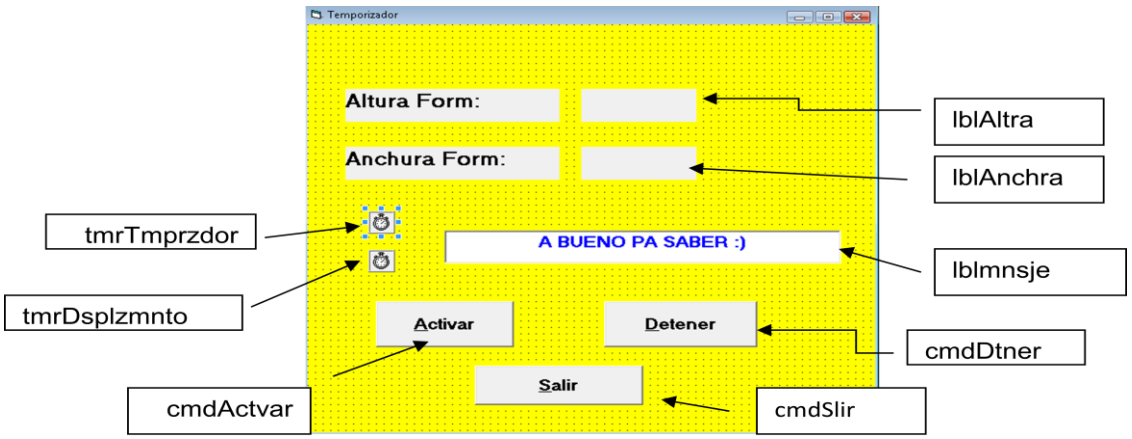
Controles: *Timer, Label, Command Button*

Propiedades: *Alignment, BackColor, Caption, Enabled, ForeColor, Interval, Left, Name, Top*

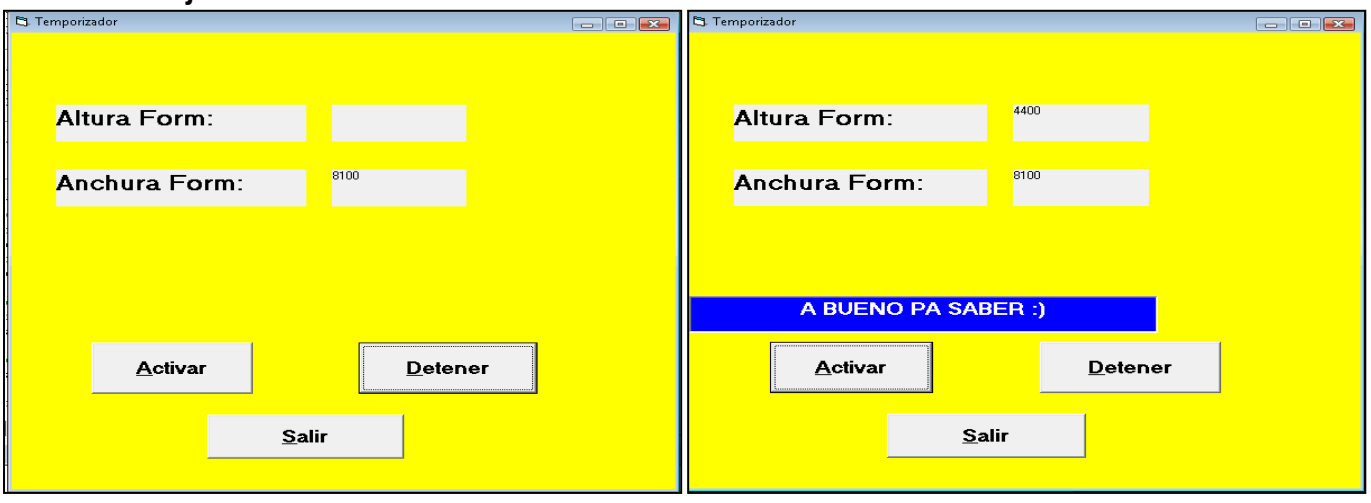
Eventos: *Click(), Load(), Timer()*

Sentencias: *If.. Then..EndIf, Variables Públicas, constantes VB de color, uso de variables indicadoras (Banderas)*

Diseño



Pantallas en ejecución.



Código

```
Public Flag, y As Integer

Private Sub cmdActvar_Click()

    tmrTmprzdor.Enabled = True

    tmrDsplzmnto.Enabled = True

End Sub


Private Sub cmdDtner_Click()

    tmrTmprzdor.Enabled = False

    tmrDsplzmnto.Enabled = False

End Sub


Private Sub cmdSlir_Click()

    End

End Sub


Private Sub Form_Load()

    Flag = 0

    'Altura del Formulario

    y = Me.Height

    'Altura del Form

    lblAnchra.Caption = Me.Height


    'Coloca etiqueta en la esquina

    'Inferior izquierda del form (eje y)

    lblmnsje.Top = y


    'Posicion en el eje x

    lblmnsje.Left = 0


    'Intervalos de los Timers

    tmrTmprzdor.Interval = 500   'Medio Segundo

    tmrDsplzmnto.Interval = 200
```

```
    tmrTmprzdor.Enabled = False
    tmrDsplzmnto.Enabled = False
End Sub

Private Sub tmrDsplzmnto_Timer()
    y = y - 100
    If y <= 0 Then
    y = Me.Height
    End If
    lblmnsje.Top = y
    lblAltra.Caption = y
End Sub

Private Sub tmrTmprzdor_Timer()
If Flag = 0 Then
    lblmnsje.BackColor = &HFFFFFF     'Blanco
    lblmnsje.ForeColor = &HFF0000     'Azul
    Flag = 1
Else
    lblmnsje.BackColor = &HFF0000    'Azul
    lblmnsje.ForeColor = &HFFFFFF    'Blanco
    Flag = 0
End If
End Sub
```

****

Aplicación 8. Contraseña

La aplicación recibirá una contraseña de máximo 10 caracteres y los cuales no se deben visualizar en la caja de texto, en su lugar deben aparecer *. Dará tres intentos para capturar contraseña correcta, de lo contrario avisa que se terminaron sus oportunidades y termina la ejecución del programa. En cada intento erróneo debe dar un aviso del error en un cuadro de mensaje y mostrar una imagen alusiva al error. Si la contraseña es correcta, igualmente dar el aviso y mostrar la imagen alusiva a que es correcto.

Se debe utilizar un solo control de Imagen, en la cual se cargará la imagen correspondiente a través de código y no de la ventana de propiedades.

Temas:

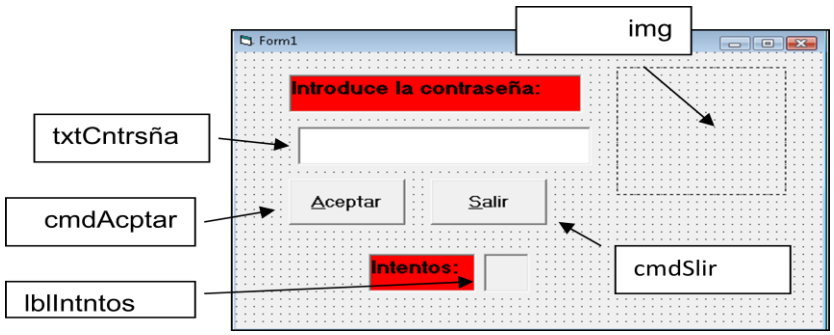
Controles: *Command Button, Label, TextBox, Image*

Propiedades: *Alignment, Caption,* ***MaxLength****, Name,* ***PasswordChar****, Picture, Text*

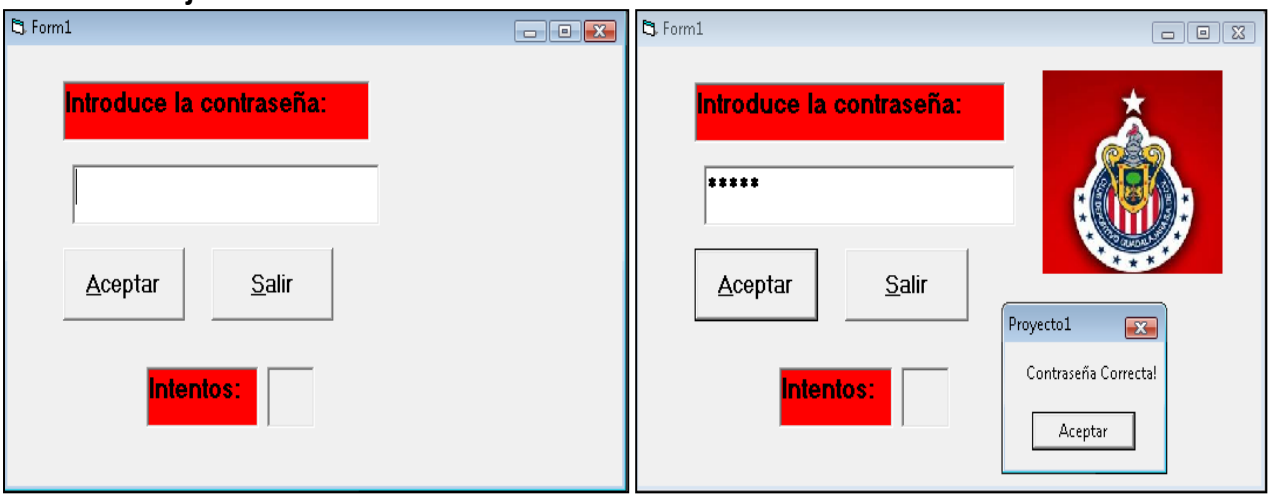
Eventos: *Click(), Load(),* ***GotFocus****()*

Sentencias: *If.. Then.. Else anidado, Variables Públicas, uso de variables indicadoras (Banderas), LoadPicture, Cuadros de Mensaje MsgBox, End, Unload,*

Diseño



Pantallas en ejecución.



Código

```
Public Contraseña, ImgAcierto, ImgError As String
Public Intentos As Integer

Private Sub cmdAcptar_Click()
    If txtCntrsña.Text = Contraseña Then
       img.Picture = LoadPicture(ImgAcierto)
       MsgBox "Contraseña Correcta!"
    Else
       Intentos = Intentos + 1
       lblIntntos.Caption = Intentos
       img.Picture = LoadPicture(ImgError)

       If Intentos > 3 Then
         MsgBox "Agoto su numero de oportunidades"
         End
         End If
    End If
End Sub

Private Sub cmdSlir_Click()
    End
End Sub

Private Sub Form_Load()
    Contraseña = "Monse"
    Intentos = 0
    ImgAcierto = App.Path & "\acierto.jpg"
    ImgError = App.Path & "\error.jpg"
End Sub
```

COMENTARIO PERSONAL

Pues me gusto este parcial por que vimos el concepto de la aplicación de Visual Basic, vimos los programas a realizar en la aplicación, todo prácticamente me pareció sencillo, lo único que se complica es la realización de códigos, donde hay veces que me desplegaban en pantalla error, puedo sugerir que me explique mas sobre los códigos para que no vuelvan a generar esos errores, pues este parcial lo sentí regular, espero poder adquirir mas conocimientos en el siguiente y ultimo parcial