

Ejercicio: 4A03E10_SYDNEY_2000

Enunciado: En el **diseño** (formulario de la izquierda), sólo estarán habilitados los botones de los países y las cajas de texto serán únicamente de lectura no podrá escribirse en ellas. En tiempo de **ejecución**, al pulsar los botones de los países sacarán el número de medallas conseguidas y se deshabilitará ese botón. Cuando se hayan pulsado los botones de los cuatro países se habilitará el botón “campeón”. Al pulsar el botón de campeón, mostrará el nombre del campeón y se habilitará el botón de salir.

Si entramos en la caja de texto de campeón se limpiarán las cajas de los países y volveremos al estado inicial. Probar hacerlo con los eventos **MouseEnter, MouseLeave, MouseMove**.

Eventos de ratón

Los eventos del mouse que simplemente notifican cuando el puntero del mouse **entra** o **sale** de los límites de un control, envían un **EventArgs** al controlador de eventos.

MouseEventArgs proporciona información sobre el estado actual del mouse, incluida la ubicación del puntero del mouse en coordenadas de cliente, qué botones del mouse se presionan y si se ha desplazado la rueda del mouse.

```
Private Sub btSalir_MouseEnter(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btSalir.MouseEnter
    Me.btSalir.BackColor = Color.Cyan

    AddHandler btAcoplado.Click, AddressOf tituloFormulario    'Añade
un controlador al evento
End Sub

Private Sub btSalir_MouseLeave(ByVal sender As Object, ByVal e As
EventArgs) Handles btSalir.MouseLeave
    btSalir.ResetBackColor()
End Sub

'al mover el ratón sobre el control, en este caso sobre el formulario, se
produce el evento
Private Sub Form1_MouseMove(ByVal sender As Object, ByVal e As
MouseEventArgs) Handles Me.MouseMove

    'e.X y e.Y: me da las coordenadas de donde se posiciona el ratón
    Me.Text = "Coordenadas raton X: " & e.X & " Y: " & e.Y

End Sub
```

[Más información sobre eventos del mouse](#)

Ejercicio:4A30E20_TEXTBOX_MULTILINEA

Enunciado: Por **diseño** crear tres etiquetas (Label) que llevarán el texto indicado, crear dos caja de texto (TextBox), la correspondiente a la etiqueta de Texto declararla multilinea. Cuatro botones con los literales correspondientes. Hay 4 etiquetas más en la parte baja del formulario, 2 de ellas con un literal concreto y las otras 2 las visualizaremos en tiempo de ejecución. **En perreación,** Al pulsar el botón Aceptar el texto escrito en ctPalabra lo inserta en ctTexto, si sigo escribiendo en ctPalabra, con Aceptar lo añade en ctTexto detrás, en la misma línea. Si pulso el botón Intro, borra la caja de texto ctPalabra. El botón Limpiar borra el contenido de texto en ctTexto. Con botón "Salir", abandonamos el programa

Al mover el ratón en ctPalabra, me contará el numero de caracteres que hay y me dirá cuantos de ellos están seleccionados, visualizándolos en las etiquetas indicadas

Añadir texto en control

```
ctTexto.Text = ctTexto.Text & " " & ctPalabra.Text  
ctTexto.AppendText(ctPalabra.Text)
```

Borrar el contenido de la caja de Texto

```
ctTexto.Clear()  
ctPalabra.Text = Nothing  
ctTexto.Text = ""
```

Contar caracteres en un TextBox

```
ctPalabra.TextLength 'núm  
ero de caracteres en el control  
etCaracteres.Text = Cstr(ctPalabra.TextLength)
```

```
ctPalabra.SelectionLength 'número de caracteres seleccionados en el control
```

```
etSeleccionados.Text = Cstr(ctPalabra.SelectionLength)
```

```
ctPalabra.SelectionStart = 5 posiciona el cursor dentro del control  
ctPalabra.SelectionLength = ctPalabra.TextLength'selecciona a partir del  
cursor los caracteres que le digamos
```

Seleccionar todos los caracteres del TextBox

```
ctPalabra.SelectAll()
```

Posiciona el foco en el control

```
ctPalabra.Focus()
```

EJERCICIO:4A03E12_COLECCIONES_CONTROLES



Enunciado: - Por **diseño**, Crea un formulario con las etiquetas y cajas de texto y botones como el formulario de la izquierda. En tiempo de **ejecución**, al cargarse el formulario inhabilitaremos las cajas de texto, al pulsar cada uno de los botones, se ejecutará la acción indicada en él. Todo ello lo haréis haciendo uso de la propiedad Controls del formulario. Al pulsar el botón Salir, cerramos la ejecución.

Recorriendo los controles de un formulario

La propiedad **Controls** de la clase Form, devuelve un tipo **ControlCollection**, que como indica su nombre, consiste en una colección que contiene los controles del formulario. Se trata de una matriz unidimensional de objetos **Control**, que es la clase base para los controles.

Podemos recorrer en iteración todos los controles de un formulario con un **For... Next**, **For Each...Next**, o cualquier otra repetitiva

Controls.Count Devuelve el número de controles en el formulario

CÓMO preguntar por el tipo de un control

```
If (TypeOf Controls(x) Is TextBox) Then . . .
```

La expresión TypeOf...Is nos permite comprobar la compatibilidad de tipo de dos variables de referencia a objeto

```
If (Controls(x).GetType Is ctTexto.GetType) Then . . .
```

'El método GetType compara dos referencias a objetos para ver si hacen referencia a instancias del mismo tipo

```
For x = 0 To Controls.Count - 1
```

```
    If (TypeOf Controls(x) Is TextBox) Then 'si es del tipo caja de texto
```

```
        Controls(x).Enabled = False Puedo acceder a todas las propiedades del control
```

```
    End IfNext
```

Extra: al hacer clic en el botón “cajas verdes” eliminaremos un TextBox y añadiremos una etiqueta nueva en el formulario

Crear un control mediante código

'como crear un control mediante código en tiempo de ejecución

```
Dim etNueva As Label = Nothing
```

'la declaro

```
etNueva = New Label()
```

'la creo

```
etNueva.Name = "etNueva"
```

'le doy un nombre

```
etNueva.Text = "Etiqueta Nueva...."
```

'le doy las propiedades

```
etNueva.Location = New Point(30, 240)
```

'le doy las propiedades

```
Controls.Add(etNueva)
```

'la añado al formulario

Eliminar un control

```
Controls.Remove(ctNombre)
```

RadioButton y GroupBox

Los controles RadioButton nos permiten definir conjuntos de opciones autoexcluyentes, de modo que situando varios controles de este tipo en un formulario, sólo podremos tener seleccionado uno en cada ocasión.

Vamos a crear un proyecto de ejemplo con el nombre **PruebaRadio Button**, en el que situaremos dentro de un formulario, una serie de controles RadioButton y un TextBox, de modo que mediante los RadioButton cambiaremos el tipo de fuente y color del cuadro de texto. Diseño inicial del formulario:

Al ejecutar el proyecto, sin embargo, no podemos conseguir establecer simultáneamente un tipo de letra y color, puesto que al pulsar cualquiera de los botones de radio, se quita el que hubiera seleccionado previamente.



Ejercicio: PruebaRadioButton

Para solucionar este problema, disponemos del control **GroupBox**, que nos permite, como indica su nombre, agrupar controles en su interior, tanto **RadioButton** como de otro tipo, ya que se trata de un control contenedor. Una vez dibujado un **GroupBox** sobre un formulario, podemos arrastrar y soltar sobre él, controles ya existentes en el formulario, o crear nuevos controles dentro de dicho control. De esta forma, podremos ya, en este ejemplo, seleccionar más de un **RadioButton** del formulario, como vemos en la Figura



El evento **CheckedChanged**, al igual que ocurría con los controles **CheckBox**, será el que tendremos que escribir para ejecutar el código en respuesta a la pulsación sobre un control **RadioButton**.

Para cambiar el tipo de fuente, instanciamos un objeto **Font** y lo asignamos a la propiedad **Font** del **TextBox**; mientras que para cambiar el color, utilizamos la estructura **Color** y la propiedad **BackColor**, también del **TextBox**.

```
txtNombre.Font = New Font("Tahoma", 12)
txtNombre.BackColor = Color.Green
```

ListBox

Un control **ListBox** contiene una lista de valores, de los cuales, el usuario puede seleccionar uno o varios simultáneamente. Entre las principales propiedades de este control, podemos resaltar las siguientes.

- **Items.** Contiene la lista de valores que visualiza el control. Se trata de un tipo `ListBox.ObjectCollection`, de manera que el contenido de la lista puede ser tanto tipos carácter, como numéricos y objetos de distintas clases.
- **Sorted.** Cuando esta propiedad contiene el valor `True`, ordena el contenido de la lista. Cuando contiene `False`, los elementos que hubiera previamente ordenados, permanecen con dicho orden, mientras que los nuevos no serán ordenados.
- **IntegralHeight.** Los valores de la lista son mostrados al completo cuando esta propiedad contiene `True`. Sin embargo, al asignar el valor `False`, según el tamaño del control, puede que el último valor de la lista se visualiza sólo en parte.
- **MultiColumn.** Visualiza el contenido de la lista en una o varias columnas en función de si asignamos `False` o `True` respectivamente a esta propiedad.
- **SelectionMode.** Establece el modo en el que vamos a poder seleccionar los elementos de la lista. Si esta propiedad contiene **None**, no se realizará selección; **One**, permite seleccionar los valores uno a uno; **MultiSimple** permite seleccionar múltiples valores de la lista pero debemos seleccionarlos independientemente; por último, **MultiExtended** nos posibilita la selección múltiple, con la ventaja de que podemos hacer clic en un valor, y arrastrar, seleccionando en la misma operación varios elementos de la lista.
- **SelectedItem.** Devuelve el elemento de la lista actualmente seleccionado.
- **SelectedItems.** Devuelve una colección `ListBox.SelectedObjectCollection`, que contiene los elementos de la lista que han sido seleccionados.
 - **SelectedIndex.** Informa del elemento de la lista seleccionado, a través del índice de la colección que contiene los elementos del `ListBox`.

Ejercicio: Listbox1



Ayuda: Cómo buscar un valor en la lista

El método `FindString()` de la lista busca un valor

```
iPosicion = lstValores.FindString("texto")
```

El campo `NoMatches` indica si no existe el valor buscado

```
If iPosicion = ListBox.NoMatches Then
    MessageBox.Show("No existe el valor")
```

```
Else
    ' si encontramos el valor en la lista,
    ' lo seleccionamos por código
End If
```

ComboBox

El ComboBox es un control basado en la combinación (de ahí su nombre) de dos controles que ya hemos tratado: TextBox y ListBox.

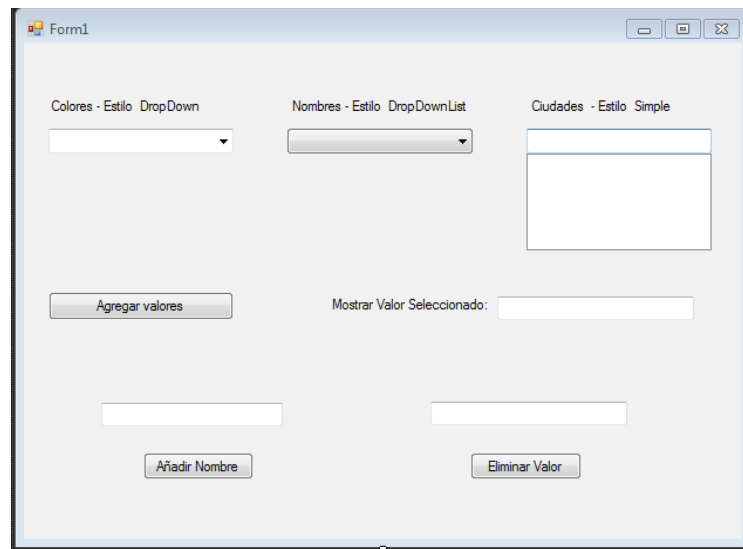
Un control ComboBox dispone de una zona de edición de texto y una lista de valores, que podemos desplegar desde el cuadro de edición.

El estilo de visualización por defecto de este control, muestra el cuadro de texto y la lista oculta, aunque mediante la propiedad **DropDownStyle** podemos cambiar dicho estilo.



La propiedad **DropDownStyle** también influye en una diferencia importante de comportamiento entre el estilo DropDownList y los demás, dado que cuando creamos un ComboBox con el mencionado estilo, el cuadro de texto **sólo podrá mostrar información**, no permitiendo que esta sea modificada. En el caso de que la lista desplegable sea muy grande, mediante la propiedad **MaxDropDownItems**, asignaremos el número de elementos máximo que mostrará la lista del control. El resto de propiedades y métodos son comunes con los controles TextBox y ListBox.

Ejercicio: PruebaComboBox



Botón Agregar valores: inicializa los valores de todos los comboBox por código:

```
ComboBox1.Items.Add("azul")
ComboColores.Items.AddRange(New String() {"AZUL", "VERDE",
"AMARILLO", "ROJO", "BLANCO", "MARRÓN", "GRANATE"})
```

Botón Eliminar valor: dado un valor lo elimina del comboBox

Eliminar un valor con index 0:

```
ComboBox1.Items.RemoveAt(0)
```

Eliminar el elemento que está seleccionado

```
ComboBox1.Items.Remove(ComboBox1.SelectedItem)
```

Eliminar el item "Azul"

```
ComboBox1.Items.Remove("Azul")
```

Botón Añadir nombres: añade un nombre al comboBox. Si el valor ya existe no lo añade

Botón Mostrar valor seleccionado: se selecciona en el comboBox el valor que se ha introducido

Menús - MenuStrip

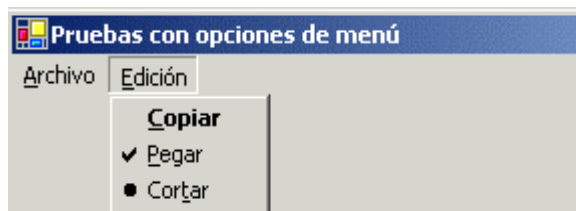
Una barra de menús es un objeto de la clase **MenuStrip** del espacio de nombres **Windows.Forms**. **MenuStrip** es un contenedor al que se le pueden añadir objetos (menús de la barra, submenús o los elementos de los menús) Podemos destacar las siguientes propiedades.

- **Text** Contiene una cadena con el literal o texto descriptivo de la opción de menú.
- **Enabled**. Permite habilitar/deshabilitar la opción de menú. Cuando se encuentra deshabilitada, se muestra su nombre en un tono gris, indicando que no puede ser seleccionada por el usuario.

- **DefaultItem.** Permite establecer opciones por defecto. En una opción de menú por defecto, su texto se resalta en negrita.
- **Checked.** Marca/desmarca la opción. Cuando una opción está marcada, muestra junto a su nombre un pequeño símbolo de verificación
- **RadioCheck.** En el caso de que la opción de menú se encuentre marcada, si asignamos True a esta propiedad, en lugar de mostrar el símbolo de verificación estándar, se muestra uno con forma de punto.
- **ShortcutKeys.** Se trata de un atajo de teclado, o combinación de teclas que nos van a permitir ejecutar la opción de menú sin tener que desplegarlo. Al elegir esta propiedad, aparecerá una lista con todos los atajos disponibles para asignar.
- **ShowShortcut.** Permite mostrar u ocultar la combinación de teclas del atajo de teclado que tenga asignado una opción de menú.
- **Visible.** Muestra u oculta la opción de menú.

Una vez finalizada la fase de diseño del menú, debemos proceder a escribir el código para sus opciones. El evento **Click** es el que permite a un control MenuItem ejecutar código cuando la opción de menú es seleccionada. Abriendo por tanto, el menú desde el diseñador del formulario, y haciendo doble clic en la opción correspondiente, nos situaremos en el editor de código, dentro del procedimiento manipulador del evento Click para esa opción.

Ejercicio:



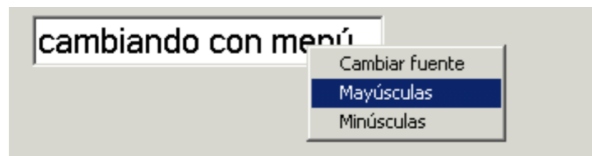
Realiza un menú como el de la imagen y añadele un TextBox donde el usuario introducirá el texto a copiar/cortar/pegar.

Cómo cortar y pegar: el portapapeles

```
Clipboard.SetText(textBox.SelectedText)
'Clipboard: proporciona métodos para poner datos en el portapapeles del sistema y
recuperarlo. SetText: agregar al portapapeles datos de texto
textBox.SelectedText = "" 'así elimino el texto seleccionado
Clipboard.SetText(textBox.SelectedText)
'el texto seleccionado lo pega al portapapeles
textBox.SelectedText = Clipboard.GetText
'GetText: recupera del portapapeles datos de texto
Clipboard.Clear()
'vacía el portapapeles
```

Menús - ContextMenuStrip

El control **ContextMenuStrip** representa un menú contextual o flotante. Este tipo de menú se asocia al formulario o a uno de sus controles, de modo que al hacer clic derecho, se mostrará sobre el elemento al que se haya asociado.



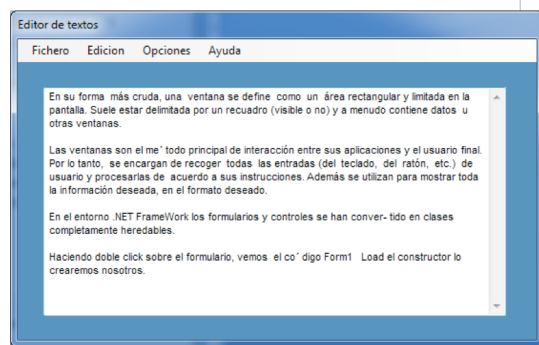
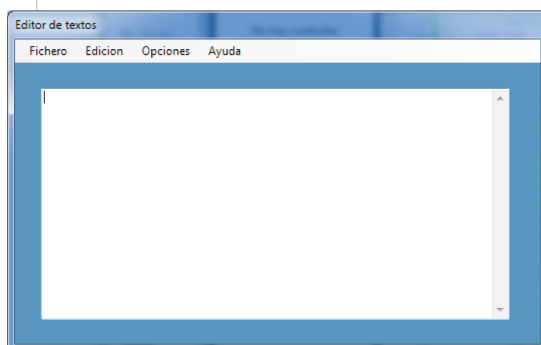
El modo de añadir un control ContextMenu y sus correspondientes opciones al formulario, es el mismo que para un MenuStrip; situándose también una referencia del menú contextual en el panel de controles especiales del diseñador. **Para asociar este menú con un control o formulario**, utilizaremos la propiedad **ContextMenuStrip** de que disponen la mayoría de los controles. En este ejemplo, insertaremos el control txtValor, de tipo TextBox, y le asociaremos el menú de contexto que acabamos de crear.

Ejercicio: ContextMenuStrip

Modifica el ejercicio anterior y añade un menú contextual a la caja de texto. Las opciones del menú contextual serán las siguientes:

- Cambiar fuente (escoge tres fuentes y crea tres opciones)
 - fuente1
 - fuente2
 - fuente3
- Mayúsculas: convierte el texto del textbox en mayúsculas `Text.ToUpper`
- Minúsculas: convierte el texto del textbox en minúsculas `Text.ToLower`

Ejercicio:4A08E20_MENU_EditorTextos



Crea un menú con los item descritos a continuación:

Fichero	Edición	Opciones				Ayuda
Salir	Cortar Copiar Pegar	Texto	Fuente	Courier New V Ariel Symbol		A cerca De...
			Tamaño	V 10 12 24		
			Colores	Fondo	V Blanco Verde Azul	
				Texto	V Negro Verde Azul	

Crear un campo multilínea con barra de scroll (ScrollBars = Both) y sin texto del tamaño de la ventana

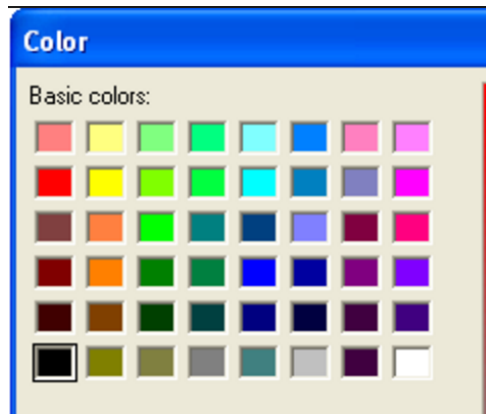
```
MEPegar.Enabled = False
MECortar.Enabled = False
MOTFCourier.Checked = True
'me permite habilitar o no items del menú
```

```
Form2.Show()
'Show() así se muestra como una vtan de dialogo no modal, no será necesario cerrar
para continuar
Form2.ShowDialog()
'ShowDialog() así se muestra como una ventana de dialogo modal, que será necesario
cerrar para continuar
tbEntrada.Font = New Font("Courier New", tbEntrada.Font.Size)
'tbEntrada.Font: obtiene o establece la fuente del texto que tiene el control
'tbEntrada.Font.Size: obtiene el tamaño de este objeto
tbEntrada.Font = New Font(tbEntrada.Font.Name, 10)
'tbEntrada.Font.Name: obtiene el nombre fuente de este objeto
```

```
tbEntrada.BackColor = Color.White
'establece el fondo de la caja de texto
```

```
Me.TopMost = True
'siempre aparece encima. Obtiene o establece si un formulario debe mostrarse como
un formulario de un nivel superior
Me.Hide()
'oculta la ventana al usuario,
Me.Dispose()
'libera todos los recursos utilizados por el objeto
Me.Close()
'cierra el formulario
```

ColorDialog



```
Dim MyDialog As New ColorDialog()

' Color personalizado.

MyDialog.AllowFullOpen = True

' Para que aparezca ayuda. (falso por defecto.)

MyDialog.ShowHelp = True

' Color inicial por seguridad y evitar crasheos tontos

MyDialog.Color = Color.Black

' Si clickea ok se cambia el color del elemento(Dialog.color)

If (MyDialog.ShowDialog() = Me.DialogResult.OK) Then

    Controls(x).ForeColor = MyDialog.Color
```

[más información](#)

InputBox()

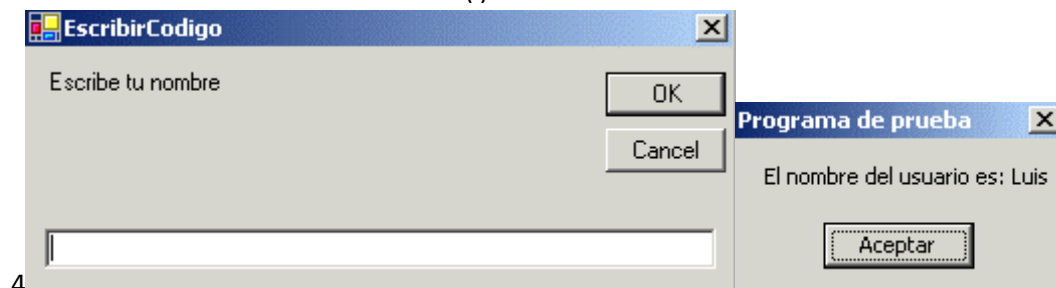
InputBox() es una función que muestra una caja de diálogo en la que el usuario puede introducir un valor, que será devuelto al aceptar dicha caja.

```
sms = InputBox(mensaje, titulo, respuestaDefecto,X,Y)
```

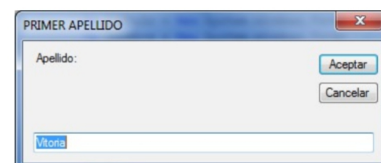
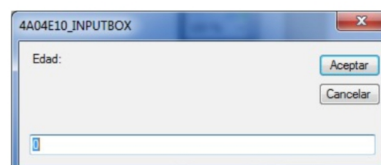
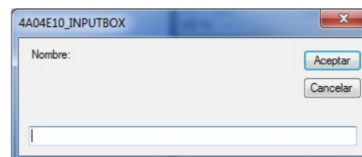
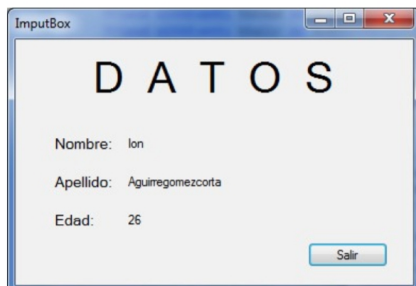
- Mensaje. Obligatorio. Cadena de caracteres con el texto que va a mostrar la caja de diálogo.
- Título. Opcional. Título que aparecerá en la caja de diálogo.
- RespuestaDefecto. Opcional. Cadena de caracteres con el valor que devolverá esta función, en el caso de que el usuario no escriba nada.
- X, Y. Opcionales. Valores numéricos que indican las coordenadas en donde será mostrada la caja. Si se omiten, se mostrará en el centro de la pantalla.
- sms: contiene la cadena de caracteres que ha introducido el usuario, si el usuario hace clic en Cancelar, se devolverá una cadena de longitud cero.

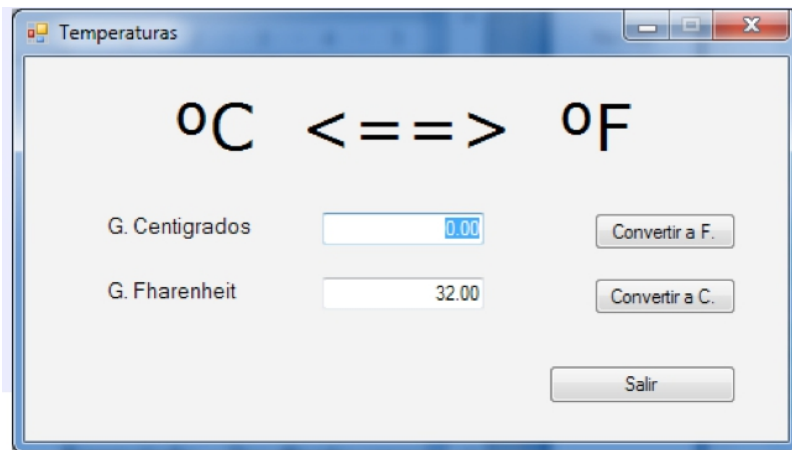
Ejercicio: PuebaInputBox

Queremos que el usuario introduzca su nombre utilizando InputBox(), volcar dicho nombre y mostrarlo usando otro MessageBox. Después de escribir su nombre en el campo de la caja, si el usuario pulsa OK, InputBox() devolverá el valor de dicho campo. Por último, mostraremos el valor de la variable usando el método Show()



4A04E10_INPUTBOX





Enunciado: Se introducirán los grados en una de las cajas de texto y al presionar el botón convertirá a °F o °C, mostrando el resultado en la caja de texto correspondiente. Al entrar en la caja de texto con el ratón, quedará seleccionado todo el contenido. Con botón “salir”, abandonamos el programa. Ten en cuenta que los caracteres introducidos deben ser números.

[Celsius a Fahrenheit](#)