

Ejercicios sobre Controles - 2.

Curso 2010-2011

Práctica 1. Añadir un menú a la aplicación

En esta práctica se creará una aplicación sencilla basada en Windows y se añadirá una barra de menú con la funcionalidad necesaria.

Los menús proporcionan a los usuarios un modo estructurado para acceder a las órdenes y herramientas que contiene una aplicación.

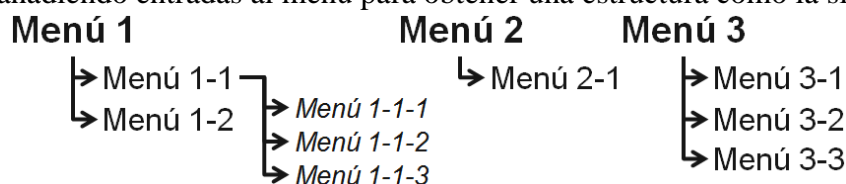
Iniciar una nueva aplicación en Visual Basic .NET

1. Abra Visual Studio .NET.
2. Añada un nuevo proyecto de tipo **Windows Form**.
3. Asígnele un nombre al proyecto y guárdelo en una carpeta del ordenador. Recuerde guardar los cambios cada cierto tiempo.
4. Haga clic en el formulario para seleccionarlo y establezca las siguientes propiedades:

Propiedad	Nuevo Valor
Name	frmPrincipal
Text	Práctica 2-1

Añadir una barra de menú

1. Añada un control **MenuStrip**.
Observe que el control queda situado debajo del formulario y que en la zona superior del formulario aparece una barra de un color diferente.
2. Haga clic en la zona izquierda de la barra superior que se muestra en el formulario.
3. Escriba la cadena **Menú 1**.
Al escribir **Menú 1**, lo que ha hecho es añadir una entrada al menú. Observe que en la medida que se añaden entradas, aparecen zonas para añadir nuevas entradas, al mismo nivel o en niveles inferiores.
4. Continúe añadiendo entradas al menú para obtener una estructura como la siguiente:



Los tipos de letras diferentes solo se utilizan en esta figura para ayudar a entender la jerarquía de las opciones del menú.

5. Haga doble clic en la opción **Menú 1** para añadir un manejador para el evento **Click** a esta entrada. Añada la siguiente línea de código al procedimiento generado:

```
MessageBox.Show("Activación del Menú 1")
```

6. Haga doble clic en la opción **Menú 1-2** para añadir un manejador al evento **Click** de esta entrada. Añada la siguiente línea de código al procedimiento generado:

```
MessageBox.Show("Activación del Menú 1-2")
```

7. Haga doble clic en la opción **Menú 1-1-1** para añadir un manejador al evento **Click** de esta entrada. Añada la siguiente línea de código al procedimiento generado:

```
MessageBox.Show("Activación del Menú 1-1-1")
```

- Haga doble clic en la opción **Menú 3-1** para añadir un manejador al evento **Click** de esta entrada. Añada la siguiente línea de código al procedimiento generado:

```
MessageBox.Show("Activación del Menú 3-1")
```

- Ejecute la aplicación y compruebe la funcionalidad activando las opciones del menú a las que se hace referencia en los apartados anteriores.
Observe que en el caso de la entrada **Menú 1**, además de ejecutar el manejador del evento lanzado, también permite acceder a las entradas que están por debajo en el orden jerárquico.
- Haga clic en la entrada **Menú 1-2** para seleccionarla. Establezca las siguientes propiedades:

Propiedad	Nuevo Valor
ShortcutKey	Ctrl-A
ToolTipText	“Esta es la entrada 1-2”

- Ejecute la aplicación y compruebe la funcionalidad de las últimas modificaciones realizadas. Primero pulse la combinación de teclas **Ctrl + A** y a continuación active la entrada **Menú 1** y coloque el puntero del ratón, sin hacer clic, sobre la entrada **Menú 1-2**.

Práctica 2. Diseñar una aplicación de múltiples documentos (MDI)

En esta práctica se creará una aplicación sencilla basada en Windows y se añadirá una barra de menú con la funcionalidad necesaria.

Los menús proporcionan a los usuarios un modo estructurado para acceder a las órdenes y herramientas que contiene una aplicación.

Iniciar una nueva aplicación en Visual Basic .NET

- Abra Visual Studio .NET.
- Añada un nuevo proyecto de tipo **Windows Form**.
- Asígnele un nombre al proyecto y guárdelo en una carpeta del ordenador. Recuerde guardar los cambios cada cierto tiempo.
- Haga clic en el formulario para seleccionarlo y establezca las siguientes propiedades:

Propiedad	Nuevo Valor
Name	frmPrincipal
Text	Práctica 2-2
Size	585; 425
IsMDIContainer	True

Añadir una barra de menú

- Siguiendo el procedimiento explicado en la práctica 1, agregue una barra de menú con una entrada que tenga como texto **Nuevo** y otra que tenga como texto **Cerrar**.

Añadir un formulario

- Agregue otro formulario a la aplicación. Utilice el nombre Secundario.vb en la ventana correspondiente. Establezca las siguientes propiedades:

Propiedad	Nuevo Valor
Name	frmSecundario
Text	Secundario
Size	210; 185

MaximizeBox	False
MinimizeBox	False

- Coloque un botón en el centro del formulario y póngale por nombre **btnUnico**.

Añadir los manejadores de eventos al formulario principal

- Copie la siguiente línea en el manejador del evento **Click** para la entrada **Cerrar** del menú:

```
Me.Close()
```

- Copie la siguiente línea en el manejador del evento **Click** para la entrada **Nuevo** del menú:

```
Dim frm As frmSecundario  
  
frm = New frmSecundario()  
frm.MdiParent = Me  
frm.Text &= " (" & count.ToString & ")"  
frm.Show()  
count += 1
```

- Añada una variable pública a la clase correspondiente al formulario. Para ello escriba la siguiente línea después del encabezamiento de la clase.

```
Public count As Integer = 0
```

- Añada el siguiente código para el manejador del evento Click del botón **btnUnico**.

```
frmPrincipal.count -= 1  
Me.Close()
```

Comprobar la funcionalidad

- Ejecute la aplicación y compruebe la funcionalidad de la misma. Observe la diferencia entre los formularios que se muestran en esta aplicación y los de las aplicaciones anteriores.

Práctica 3. Validar la entrada de datos

En esta práctica se creará una aplicación basada en Windows para la conversión de temperaturas.

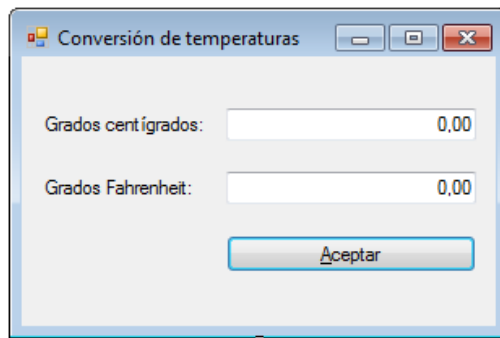
Iniciar una nueva aplicación en Visual Basic .NET

- Abra Visual Studio .NET.
- Añada un nuevo proyecto de tipo **Windows Form**.
- Asígnele un nombre al proyecto y guárdelo en una carpeta del ordenador. Recuerde guardar los cambios cada cierto tiempo.
- Haga clic en el formulario para seleccionarlo y establezca las siguientes propiedades:

Propiedad	Nuevo Valor
Name	frmPrincipal
Text	Conversión de Temperaturas
Size	310; 205

Añadir otros controles al formulario

1. Añada al formulario dos etiquetas, dos cajas de texto y un botón. El aspecto que debe tener el formulario se muestra a continuación:



2. Establezca las siguientes propiedades a los controles:

Control	Propiedad	Nuevo Valor
Textbox1 (Superior)	Name	txtGradosC
	Text	0,00
	TextAlign	Rigth
Textbox2 (Inferior)	Name	txtGradosF
	Text	32,00
	TextAlign	Rigth
Button1	Name	btnAceptar
	UserMnemonic	True
	Text	&Aceptar
Form1	AcceptButton	btnAceptar

3. Añada el siguiente código al manejador del evento **Click** del botón:

```
Dim grados As Double
```

```
grados = Convert.ToDouble(txtGradosC.Text) * 9.0 / 5.0 + 32.0
' Mostrar el resultado redondeado a dos decimales
txtGradosF.Text = String.Format("{0:F2}", grados)
```

Comprobar funcionalidad

1. Ejecute la aplicación.
2. Compruebe su funcionalidad. Escriba un número entero o real en la caja de texto superior y haga clic en el botón.
3. Escriba otro número y presione la tecla **Enter**.
4. Escriba otro número y presione las teclas **Alt+A**.
5. Escriba una letra en la caja de texto superior y haga clic en el botón. Intente explicarse por qué ocurre el efecto sucedido.
6. Cambie el manejador del evento **Click** del botón por el que se indica a continuación y observe lo que sucede al escribir una letra en la caja de texto de la temperatura en grados centígrados.

```
Dim grados As Double
```

```
Try
```

```
grados = Convert.ToDouble(ctGradosC.Text) * 9.0 / 5.0 + 32.0
' Mostrar el resultado redondeado a dos decimales
txtGradosF.Text = String.Format("{0:F2}", grados)
```

```
Catch ex As FormatException
txtGradosC.Text = "0,00"
```

```

        txtGradosF.Text = "32,00"
    End Try

```

7. Busque información en MSDN sobre la instrucción **Try-Catch** para determinar las ventajas de su utilización en situaciones como la anterior.

Realizar cambios en la funcionalidad

1. Sería interesante hacer que la aplicación se comporte como un conversor bidireccional, es decir, de grados Centígrados a Fahrenheit y viceversa. Para ello añada una variable de tipo **TextBox** a la clase **Form1**, tal como se muestra a continuación:

```

Private objTextBox As TextBox = Nothing

```

2. Reemplace el manejador del evento **Click** del botón por el siguiente:

```

Try
    Dim grados As Double
    ' Si se escribió en la caja de texto grados centígrados
    If (objTextBox Is txtGradosC) Then
        grados = Convert.ToDouble(txtGradosC.Text) * 9.0/5.0+32.0
        ' Mostrar el resultado redondeado a dos decimales
        txtGradosF.Text = String.Format("{0:F2}", grados)
    End If
    ' Si se escribió en la caja de texto grados Fahrenheit
    ...
    If (objTextBox Is txtGradosF) Then
        grados = (Convert.ToDouble(txtGradosF.Text)-
        32.0)*5.0/9.0
        ' Mostrar el resultado redondeado a dos decimales
        txtGradosC.Text = String.Format("{0:F2}", grados)
    End If
    Catch ex As FormatException
        txtGradosC.Text = "0,00"
        txtGradosF.Text = "32,00"
    End Try

```

3. Añada a cada control **TextBox** un procedimiento manejador para el evento **KeyPress**. En cada uno de los procedimientos escriba la siguiente línea de código:

```

objTextBox = CType(sender, TextBox)

```

4. Investigue en el MSDN acerca de la funcionalidad del evento **KeyPress**.

Comprobar funcionalidad


1. Ejecute la aplicación.
2. Compruebe su funcionalidad. Escriba un número entero o real en la caja de texto superior y haga clic en el botón. Realice la misma acción en la caja inferior.

Utilizar el mismo procedimiento manejador de eventos en dos controles diferentes

Observe que el procedimiento manejador para el evento **KeyPress** es idéntico para ambos procedimientos. Su función consiste en identificar en cuál de las dos cajas de texto se ha introducido el dato a convertir y para ello asigna a la variable **objTextBox** una referencia al control que ha recibido el dato.

Para hacer más eficiente el programa se puede utilizar un único procedimiento para ambos controles y utilizar el parámetro **sender** para obtener dicha referencia. Para ello haga lo siguiente:

1. Borre el procedimiento manejador del evento **KeyPress** a los dos controles **TextBox**. Si no quiere borrarlos, también puede comentarlos para que no constituyan bloques

ejecutables. Esto se logra marcando el código de todo el procedimiento y presionando el botón 

2. En lugar de los dos procedimientos, añada el siguiente:

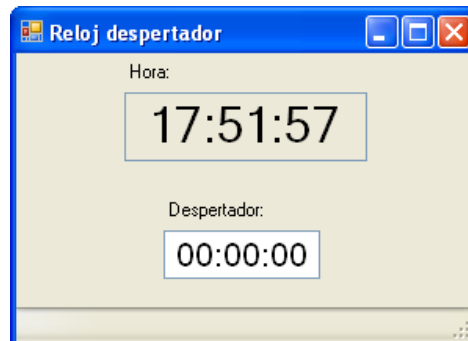
```
Private Sub KeyPress_Handler(ByVal sender As System.Object, _  
    ByVal e As System.Windows.Forms.KeyPressEventArgs) _  
    Handles txtGradosF.KeyPress, txtGradosC.KeyPress  
    objTextBox = CType(sender, TextBox)  
End Sub
```

Observe que después de la palabra **Handles** se hace referencia, separados por comas, a los eventos **KeyPress** de las dos cajas de texto.

3. Compruebe nuevamente la funcionalidad del programa.

Ejercicios

1. Implementar un reloj despertador. La aplicación tendrá el siguiente aspecto:



La hora actual se muestra en un control de solo lectura, es decir, se muestra la hora del reloj de tiempo real del ordenador y esta no se puede cambiar. En el control inferior se puede escribir, en formato HH:MM:SS, la hora a la que se quiere que se active la alarma. Dicha alarma consistirá en hacer parpadear varias veces la hora de alarma al tiempo que se emite una señal sonora.

Hay que verificar que la hora de alarma esté escrita en el formato correcto.

Sugerencias:

Utilizar el objeto **DateTime**. (Documentación en [http://msdn.microsoft.com/es-es/library/system.datetime_members\(VS.80\).aspx](http://msdn.microsoft.com/es-es/library/system.datetime_members(VS.80).aspx))

Para actualizar la hora se puede utilizar un control **Timer**.

El control **Timer** es un control que genera un evento **Tick** periódicamente a la aplicación al transcurrir un intervalo de tiempo.

Propiedades	Significado
Interval	Intervalo de tiempo (en milisegundos) que tiene que transcurrir para que se genere el evento Tick .
Enabled	Habilita el control

Métodos	Significado
Start	Inicia la temporización.
Stop	Detiene la temporización

Para convertir el formato de hora a una cadena de caracteres se puede utilizar *DateTime.Now.ToLongTimeString*.

Un control se puede hacer invisible estableciendo el valor *False* en la propiedad *Visible* del mismo.

Mediante la subrutina *Beep()* se puede emitir una señal sonora. (*Para que se escuche la señal sonora deben estar conectados los altavoces externos*)

El efecto de parpadeo se puede lograr mediante otro temporizador que se inicie cuando se haya alcanzado el tiempo de la alarma. En ese momento también se puede emitir la señal sonora.

2. Añadir al despertador dos botones: uno para activar/desactivar la alarma y el otro para detenerla, una vez que se ha disparado.

Cuando la alarma está activada, debe aparecer alguna indicación en el formulario.

Los **RadioButton** son controles del tipo botón, con la característica de que mantienen un estado (marcado o no) hasta que explícitamente se le envía la orden de cambiarlo. Esta orden puede enviarse desde el programa o mediante la interacción directa del usuario de la aplicación, al hacer clic sobre él.

Al colocar varios **RadioButton** en un formulario, estos forman un grupo, de manera que al activar uno de ellos, se desactiva el que estuviese activado con anterioridad. Esto último de manera totalmente transparente al programador. Es por ello que se utilizan en aplicaciones para seleccionar opciones mutuamente excluyentes.

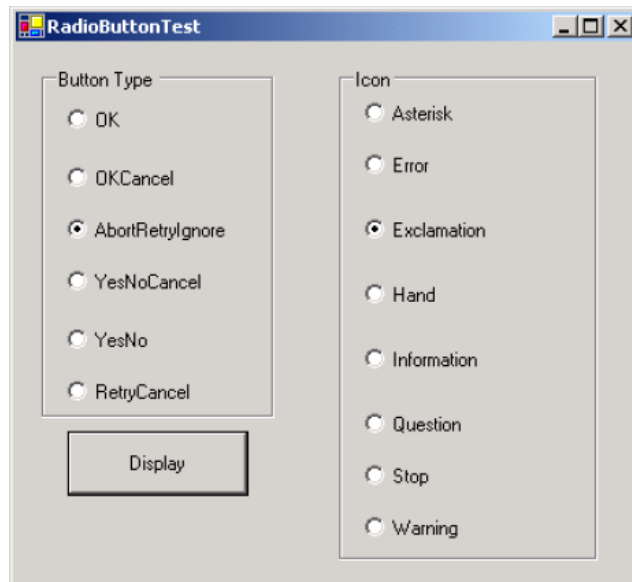
Se pueden tener varios grupos de **RadioButton** en una aplicación. Cada grupo se independiza del otro mediante el control **GroupBox**, por lo que al activar un **RadioButton** de uno de los grupos se desactiva el que estuviese activado en ese grupo pero no sucede lo mismo con los de los demás grupos.

Para establecer un grupo de **RadioButton** primero se coloca un control **GroupBox** en el formulario, el cual debe tener un tamaño adecuado para contener todos los **RadioButton**. A continuación se colocan los **RadioButton** dentro del área del **GroupBox**.

Propiedades	Significado
Checked	Indica, con valor <i>True</i> , si el control está marcado o <i>False</i> en caso contrario.
Text	Establece el texto que aparece en la parte derecha del control

Métodos	Significado
Click	Se genera al pulsar el botón izquierdo del ratón sobre el control.
CheckedChanged	(Evento por defecto) Se genera cada vez que se activa o se desactiva el control.

3. Realizar una aplicación que demuestre las diferentes posibilidades que tiene Visual Basic para mostrar ventanas de aviso mediante la función *MessageBox*. La aplicación debe tener la siguiente apariencia:



Al pulsar en el botón “**Display**”, se muestra una ventana de tipo **MessageBox** con el mensaje “Texto de Prueba” y la configuración establecida en la aplicación.

Sugerencias:

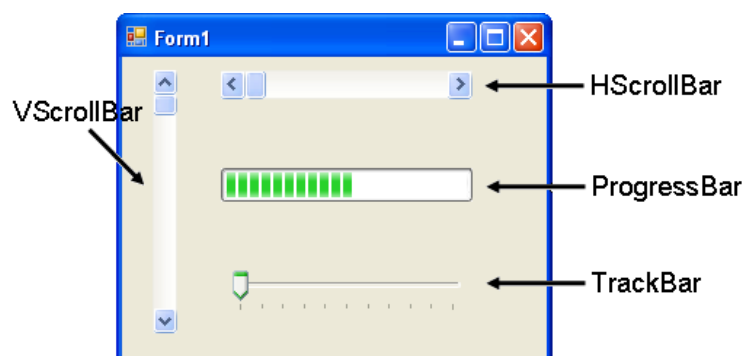
Ver en MSDN las diferentes opciones para mostrar una ventana **MessageBox**.

Utilizar dos variables globales a las que se les asigne un valor en función de las opciones marcadas. Una de las variables se actualiza de acuerdo con los botones de la columna “Button Type” y la otra se actualiza de acuerdo a los botones de la columna de la derecha. En el primer caso la variable puede tomar valores de 1 a 6 y en el segundo caso puede tomar valores de 1 a 8.

Utilizar una instrucción **Select-Case** en el manejador del evento **Click** del botón para mostrar el mensaje.

Los controles de rango definido (**VScrollBar**, **HScrollBar**, **TrackBar** y **ProgressBar**) establecen un valor entero en un rango previamente definido

En la siguiente figura se muestran estos controles.



El valor se establece al desplazar el elemento móvil, excepto en el control **ProgressBar**, que se establece al serle asignado directamente un valor a la propiedad **Value**.

Propiedades	Significado
Minimum	Indica el extremo inferior del rango de valores
Maximun	Indica el extremo superior del rango de valores
Value	Indica el valor actualmente seleccionado
SmallChange	Número de posiciones que se mueve el control

(excepto ProgressBar)	deslizante como respuesta a una acción de teclado.
LargeChange (excepto ProgressBar)	Número de posiciones que se mueve el control deslizante cuando se presiona las tecla AvPg ó RePg.
Step (solo ProgressBar)	Cantidad por la qua aumenta <i>Value</i> al emplear el método PerformStep()
TickStyle (solo TrackBar)	Define la forma del elemento móvil del control
TickFrequency (solo TrackBar)	Define el espaciado entre las marcas.

Método	Significado
ValueChanged	Tiene lugar cuando cambia el valor de Value .

4. Realizar un programa con la siguiente interfaz de usuario.

La aplicación convierte valores de temperatura en grados Centígrados (en un rango entre 0 y 100°C) a su equivalente en grados Fahrenheit.

Los valores de temperatura en grados centígrados se indican escribiendo directamente en la caja de texto correspondiente o desplazando el cursor de la barra vertical. Si se escribe la temperatura en la caja de texto, el cursor de la barra vertical debe moverse hacia el punto equivalente. De igual forma se actualiza la caja de texto si se mueve el cursor de la barra vertical para indicar una temperatura.

Debe verificarse que la entrada de datos en la caja de texto tenga el valor correcto en cuanto a los límites establecidos y al tipo de dato.

5. Modificar el programa del ejercicio anterior para que se puedan cambiar los límites de temperatura a convertir.