Ejercicios sobre Controles - 4.
Curso 2010-2011

Control *ListBox*.

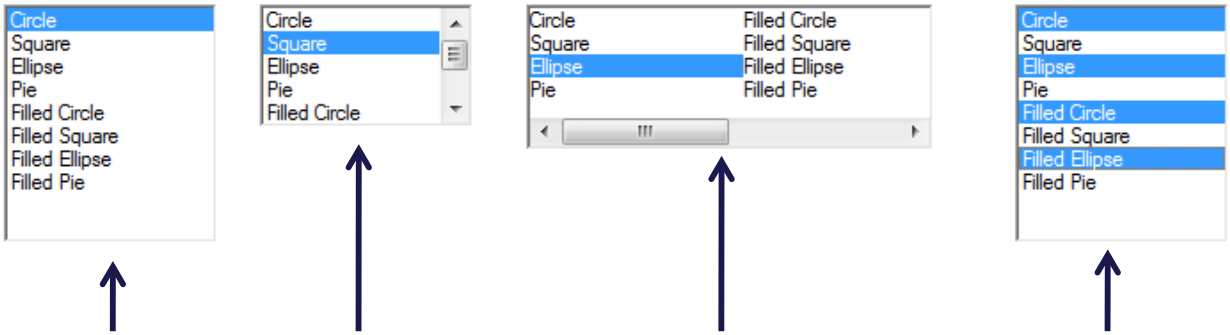
Una lista simple (*ListBox*) es un control que pone a disposición del usuario un conjunto de valores que podrá elegir haciendo clic sobre ellos. La selección de los elementos puede ser simple (uno solo) o múltiple (varios al mismo tiempo, mediante una combinación de teclas).

Los elementos de la lista se visualizan en una sola columna de forma predeterminada, aunque también se pueden visualizar en varias columnas, en función de la relación entre la cantidad de elementos y las dimensiones exteriores del control.

Cuando los elementos no pueden ser visualizados simultáneamente en el espacio disponible, aparece automáticamente una barra de desplazamiento horizontal, vertical o ambas, si fuera el caso.

Cada elemento tiene asociado un índice que depende de la posición del mismo en la lista. El índice del primer elemento es el 0.

En la figura siguiente se observan algunos ejemplos de la apariencia que puede tomar este control en una aplicación.



Propiedades de *ListBox*

Items	Es una colección que contiene todos los elementos de la lista. Es de tipo *String*.
SelectedItems	Es una colección que contiene los elementos seleccionados de la lista.
MultiColumn	Indica en True si los elementos de la lista se muestran en varias columnas.
SelectedIndices	Es una colección que contiene los índices de los elementos seleccionados de la lista.
SelectedIndex	Almacena el índice del elemento seleccionado. Si hay varios seleccionados, solo almacena el último. Tiene valor -1 cuando no hay elementos seleccionados.
SelectedItem	Almacena el elemento seleccionado. Si hay varios seleccionados, solo almacena el último.
Sorted	Indica en True si los elementos de la lista se muestran ordenados alfabéticamente.
SelectionMode	Indica si se pueden seleccionar uno o varios elementos simultáneamente.
Text	Obtiene o busca el texto del elemento actualmente seleccionado en el control.

Eventos de la colección *Items*

SelectedIndexChanged Se produce cuando ha cambiado la propiedad *SelectedIndex*.
SelectedValueChanged Se produce cuando cambia la propiedad *SelectedValue*.

Los elementos que se muestran en el control se almacenan en una **colección** (matriz). A la colección se pueden asignar valores en tiempo de diseño o en tiempo de ejecución. Como se puede observar en la tabla anterior, la colección de elementos puede ser referenciada mediante la propiedad *Items*.

Las colecciones son clases, con sus consiguientes propiedades y métodos asociados.

Métodos de la colección *Items*

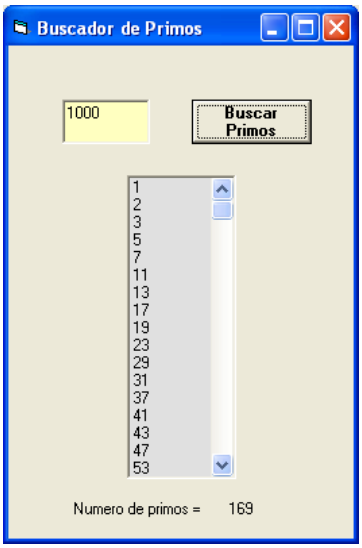
Add	Añade un elemento a la colección al final de esta
AddRange	Añade una matriz de elementos a la colección al final de esta
Insert	Inserta un elemento específico en una ubicación conocida de la colección
Remove	Elimina un elemento específico de la colección
RemoveAt	Elimina un elemento en una ubicación conocida de la colección
Clear	Elimina todos los elementos de la colección
Count	Obtiene el número de elementos de la colección

Ejemplos:

- Muestra en una caja de texto, llamada **txtPrueba**, el elemento seleccionado de una lista llamada **lbxPrueba**:
```
txtPrueba.Text = lbxPrueba.SelectedItem
```
- Muestra en una caja de texto, llamada **txtPrueba**, todos los elementos seleccionados de una lista llamada **lbxPrueba**:
```
For i = 0 To lbxPrueba.SelectedItems.Count - 1
        txtPrueba.Text &= lbxPrueba.SelectedItems(i).ToString
Next
```
- Añade el contenido de una caja de texto al final de una lista:
```
lbxPrueba.Items.Add(txtPrueba.Text)
```

Ejercicios

1. Desarrollar una aplicación con la apariencia mostrada a continuación:



En la caja de texto de fondo amarillo se escribe un número entero de tipo Long. Al presionar el botón “Buscar Primos” se muestran todos los números primos que hay entre 2 y el número antes mencionado, en la lista simple (*ListBox*) que está en el centro del formulario.
En la parte inferior de la ventana debe aparecer la cantidad de números primos que hay en la caja de texto.
Añadir la siguiente funcionalidad:

a. Mostrar los números pares menores que un elemento seleccionado.
b. Mostrar el promedio de todos los seleccionados si hay más de uno.

Debe añadir, también, los controles necesarios para cumplir con los requisitos anteriores.
Deben inhabilitarse todos los controles que no se utilicen, en dependencia de la cantidad de elementos marcados. Esto quiere decir que si se marca un solo elemento de la lista solo estarán habilitados los controles necesarios para satisfacer la funcionalidad del apartado a. Lo mismo ocurre para los controles necesarios para cumplir con la funcionalidad especificada en el apartado b.
Sugerencia:
La propiedad *Count* de la colección *SelectedItems* devuelve la cantidad de elementos seleccionados. Por ejemplo:

```
txtPrueba.Text = lbxPrueba.SelectedItems.Count
```

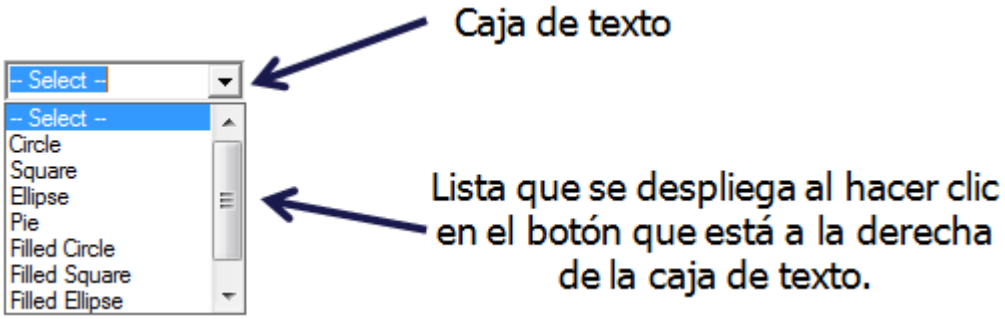
muestra en una caja de texto, llamada **txtPrueba**, la cantidad de elementos seleccionados de la lista llamada **lbxPrueba**.

2. Realice una aplicación en la que se pueda marcar un número en una lista simple que aparece en la ventana. Dicha lista contiene todos los números enteros entre 3 y 100 (ambos incluidos). El usuario podrá marcar un número de la lista y la aplicación mostrará un mensaje si el número es primo o no. Solo se puede marcar un número de cada vez.
   Sugerencia:
   Realizar el llenado de la lista de forma automática en el evento FormLoad, mediante la propiedad *Add* de la colección *Items*. Por ejemplo:

```
For i = 3 to 100
  lbxNumeros.Items.Add(i.ToString)
Next
```

Control *ComboBox*.

Una lista desplegable (*ComboBox*), también denominada “Cuadro Combinado”, es una combinación de una lista simple y una caja de texto que permite la selección de un solo elemento cada vez. En la siguiente figura se muestra el control y sus partes:



Hay tres estilos diferentes de listas desplegables: editable, no editable y estática.
Las propiedades indicadas en la tabla correspondiente al control *ListBox* también se aplican a la lista desplegable.

Otras Propiedades

DropDownStyle Valores posibles:

- *DropDown* (valor predeterminado): La lista es editable por lo que el usuario puede escribir información en la caja de texto de la lista o seleccionar un elemento de la misma.
- *DropDownList*: La lista no es editable por lo que el usuario solo puede seleccionar elementos de la lista.
- *Simple*: La lista es estática por lo que se mantendrá siempre desplegada.

Text Obtiene o establece el texto asociado al control.

Métodos

BeguinUpdate Se invoca antes de la inserción de elementos para que la lista se repinte cada vez que se añade un valor.

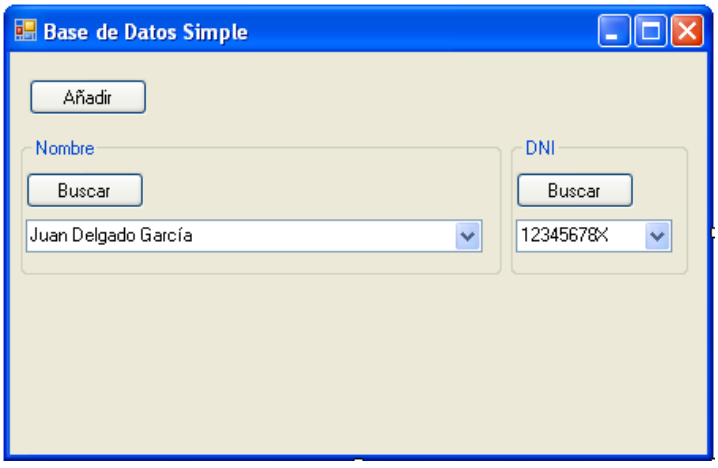
EndUpdate Se invoca cuando se añaden elementos a la lista pero se desea que estos sean mostrados cuando se inserte el último valor.

FindString Busca el primer elemento de la lista que comienza con la cadena especificada.

FindStringExact Busca el primer elemento de la lista que coincide con la cadena especificada.

Ejercicios

3. Realizar el ejercicio 2 con un *ComboBox*.
4. Realizar un programa que gestione una base de datos muy simple en la que se almacenan el nombre y el DNI de un grupo de personas.
   A continuación se muestra una sugerencia para la interfaz gráfica de usuario.



El botón “Añadir” inserta una nueva entrada en la base de datos. Esto significa que al final de cada una de las listas se añade la información correspondiente (Nombre y DNI).
En los controles *ComboBox* debe mostrarse en todo momento la información de la primera entrada a la base de datos, excepto al iniciar la aplicación, que se mostrarán en blanco ya que la base de datos está vacía.
Los botones “Buscar” permiten buscar información en las listas correspondientes. Si la búsqueda se realiza por nombre, el programa debe dar la posibilidad de entrar una cadena de texto y comprobar si dicha cadena se encuentra en la lista de nombres. Si se encuentra la cadena se muestra en la caja de texto de la lista y en la otra caja se muestra el DNI de la persona encontrada. De forma análoga ocurre con la búsqueda por DNI.

Si la información no se encuentra en cualquiera de los dos casos, se muestra un mensaje en una ventana de mensajes y se muestra la información correspondiente a la primera persona de la lista.
La búsqueda debe ser exacta en ambos casos, es decir, si no se encuentra la cadena exacta, debe mostrarse un mensaje de error.
Sugerencia: Para la entrada de datos se puede utilizar la función *InputBox* si no se quiere colocar una cantidad excesiva de controles en el formulario. Dicha función proporciona un cuadro de diálogo en el que el usuario puede escribir una cadena de caracteres. la cadena de caracteres se devuelve en el nombre de la función cuando el usuario hace clic en el botón "Aceptar". Puede encontrar más información sobre *InputBox* en http://msdn.microsoft.com/es-es/library/6z0ak68w(v=VS.90).aspx

5. Añadir a la aplicación del ejercicio anterior la posibilidad de que la información quede ordenada alfabéticamente.
Nota: Cuidar que se mantenga la relación entre nombre y DNI. Para ello puede utilizar una estructura de datos con dos campos (nombre y DNI), en la que se almacenen los elementos que se vayan añadiendo. Cada vez que se añada uno se ordena toda la estructura y posteriormente se copia cada campo a la colección *Items* de los controles correspondientes.
A continuación se muestra un procedimiento en el que se implementa una versión del algoritmo *BubbleSort*, adaptado para ordenar una matriz de estructuras, según el contenido del campo **nombre**.

```
Structure data
 nombre, DNI As String
End Structure

Private Sub SortData(ByRef target() As data)
 Dim s As Boolean = True
 Dim aux As data
 Dim i As ULong
 Dim k As ULong = 0

 If target.Length > 0 Then k = target.Length - 1
 While (s = True) And (k > 0)
    s = False
    For i = 1 To k
      If target(i - 1).nombre >= target(i).nombre Then
           aux = target(i - 1)
           target(i - 1) = target(i)
           target(i) = aux
           s = True
      End If
    Next i
    k = k - 1
 End While
End Sub
```

6.