

TegoSoft Inc.

Copyright © 1994-1996 TegoSoft Inc. ® All Rights Reserved

P.O.Box 389, Bellmore, NY 11710

Web Site: <http://www.tegosoft.com>

 e-mail: tegosoft@msn.com

Lesson 25 -

List Boxes and Clipboard Operations (Copy, Paste Cut)


In this lesson you'll learn to write programs that utilize List Boxes. In this lesson you'll also learn how to use the Clipboard for performing standard clipboard operations (Copy. Paste. Cut).


Creating the Directory of the Project and Saving the Files of the Project


As usual, you'll start by creating the directory of the project, and saving the files of the project.

 Create the **C:\VBMyProg\Lesson25** directory.


You'll save the files of this lesson to this directory.

 Select **New Project** from the **File** menu of Visual Basic.

 Make the window of Form1 the selected window, select **Save File As** from the **File** menu of Visual Basic, and save the file as **MyList.FRM** inside the **C:\VBMyProg\Lesson25** directory.

 Select **Save Project As** from the **File** menu of Visual Basic, and save the project as **MyList.VBP** inside the **C:\VBMyProg\Lesson25** directory.

Always Use Option Explicit

 Inside the general declarations section of Form1 type the following code:

```
' Force variable declarations  
Option Explicit
```

The **Option Explicit** statement forces you to declare variables before using the variables.

Designing the Window of the MyList Program

You'll design the window of the MyList program so that it will look as shown in Figure 25.1.

 Set the properties of Form1 as follows:

Name: Form1

Caption: The MyList Program

BackColor: White

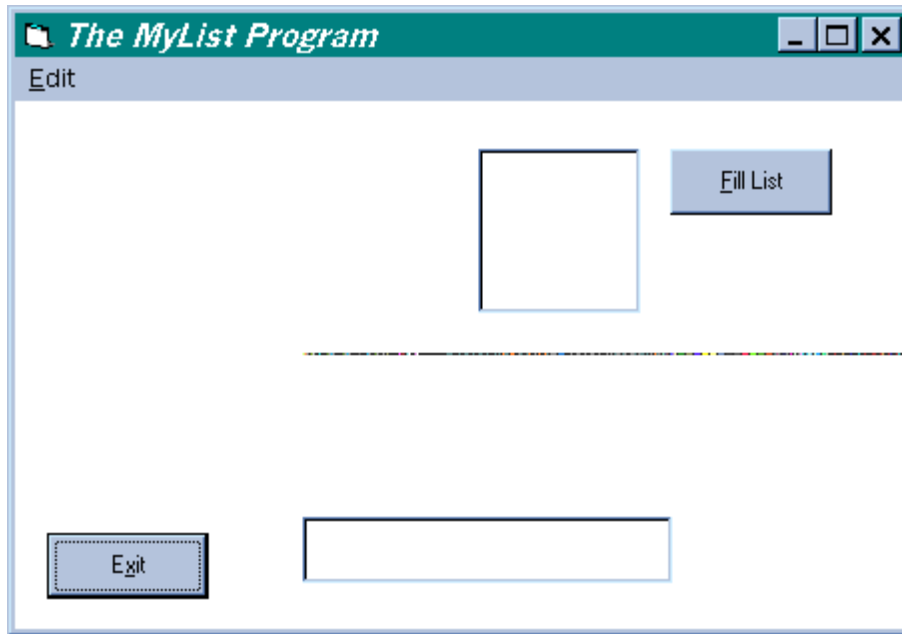



Figure 25.1. *The MyList program*

Implementing the Exit Button

You'll now implement the Exit button:


 Place a CommandButton inside Form1, then set the properties of the CommandButton as follows:

Name: cmdExit

Caption: E&xit

Attaching Code to the Click Event of the Exit Button

You'll now attach code to the **Click** event of the Exit button:

 Add the following code inside the **cmdExit_Click()** procedure:

```
Private Sub cmdExit_Click()
```

=====


End

End Sub

So whenever the user clicks the Exit button, the MyList program terminates itself.

Placing a ListBox Control

You'll now place a ListBox control inside Form1.

 Double click the ListBox control inside the Toolbox of Visual Basic.

Visual Basic responds by placing a ListBox control inside Form1.

NOTE

The icon of the ListBox control inside the Toolbox of Visual Basic is shown in Figure 25.2.

In Figure 25.2 the icon of the ListBox control is shown as the fifth row from the top on the right column. However, in your toolbox, the icon of the ListBox control may be located in a different location.

NOTE

When you place the mouse cursor (without clicking) on the ListBox icon inside the Toolbox, a yellow rectangle appears with the text **List**Box in it.

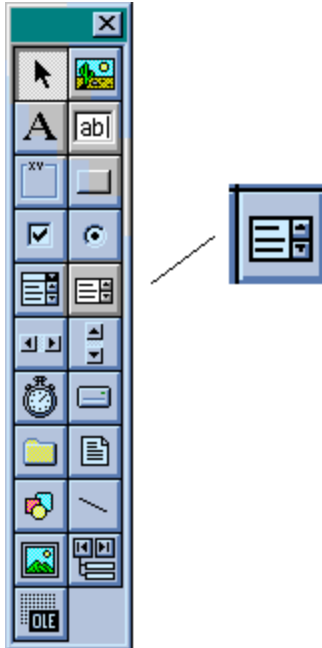





Figure 25.2. *The Toolbox of Visual Basic*

 Drag the lower edge of the list box to make its height longer.

 Make sure that the **Name** property of the ListBox control that you placed inside Form1 is **List1**

Placing the Fill List Button

You'll now place a CommandButton inside Form1. This button will serve as the mechanism to fill the **List1** list box.

 Place a CommandButton inside Form1, and set its properties as follows:

Name: cmdFillList


Caption: &Fill List

Before attaching any code to the **cmdFillList** button, let's see what you accomplished so far:

 Select **Save Project** from the **File** menu of Visual Basic to save your work.

 Execute the MyList program.

As shown in Figure 25.3, the list box has no items in it.

 Click the **Exit** button to terminate the program.

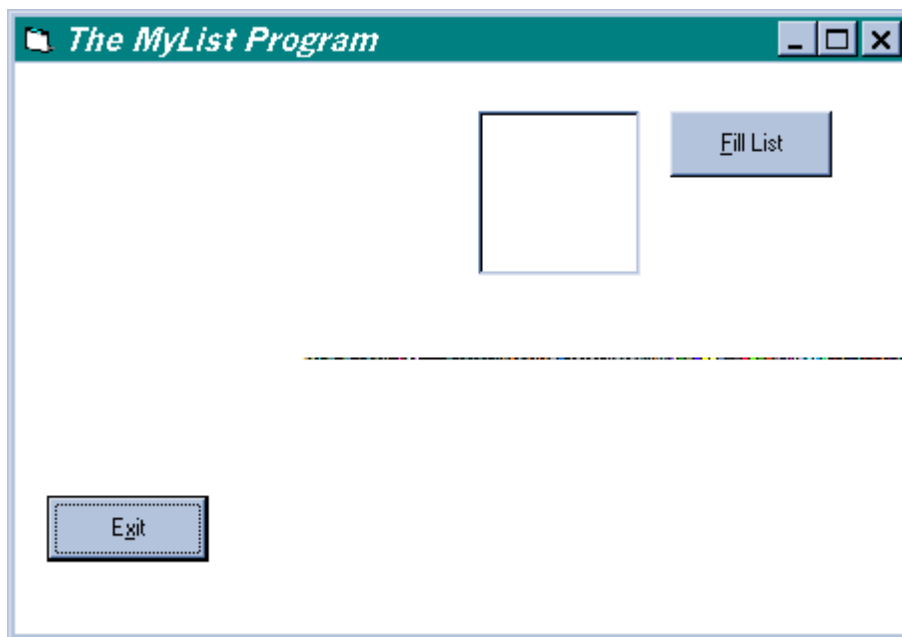



Figure 25.3. *The MyList displays an empty list box*

Filling the List Box

You'll now write code that fills the **List1** list box.

 Type the following code inside the **cmdFillList_Click()** procedure:

```
Private Sub cmdFillList_Click()  
  
Dim ItemsOfList(0 To 8)  
Dim Counter  
  
ItemsOfList(0) = 100  
ItemsOfList(1) = 101  
ItemsOfList(2) = 102  
ItemsOfList(3) = 103  
ItemsOfList(4) = 104  
ItemsOfList(5) = 105  
ItemsOfList(6) = 106  
ItemsOfList(7) = 107  
ItemsOfList(8) = 108  
  
For Counter = 0 To 8  
    List1.AddItem Str(ItemsOfList(Counter))  
Next  
  
End Sub
```

The code that you typed declares the **ItemsOfList** array as follows:

```
Dim ItemsOfList(0 To 8)
```

Also, a local variable is declared:

```
Dim Counter
```

The elements of the array are then filled with integers:

=====

```
ItemsOfList(0) = 100
ItemsOfList(1) = 101
ItemsOfList(2) = 102
ItemsOfList(3) = 103
ItemsOfList(4) = 104
ItemsOfList(5) = 105
ItemsOfList(6) = 106
ItemsOfList(7) = 107
ItemsOfList(8) = 108
```

Finally, a **For()** loop is executed to fill the elements of the list box:

```
For Counter = 0 To 8
    List1.AddItem Str(ItemsOfList(Counter))
Next
```

The **AddItem** method is used to fill the elements of the list box. Upon executing the **For()** loop for the first time, the **AddItem** method adds the first item to the list. When the **For()** loop is executed for the second time, the **AddItem** method adds the second item to the list, and so on.

Let's see your code in action:


 Select **Save Project** from the **File** menu of Visual Basic to save your work.

 Execute the MyList program.

At this point in the execution of the program, the list box is empty.

 Click the **Fill List** button.

The MyList responds by filling the items in the list (see Figure 25.4).

 Click the Exit button to terminate the MyList program.

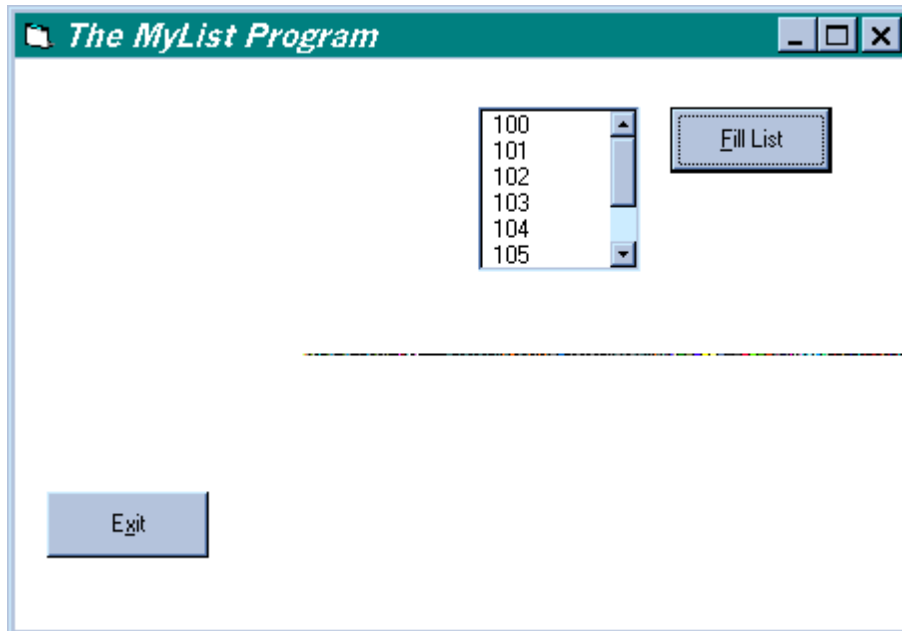


Figure 25.4. *The list box is filled with items.*

 **NOTE**

If the list box is not high enough to fit all the items, then a scrollbar is automatically attached to the list box so that the user can scroll the list. In Figure 25.4 the list box is not high enough, hence a scrollbar appears inside the list box.


Using the ListBox for Easy Selection

Ok, you placed a list box inside Form1, and you filled the list box with items. Typically, the list box control is used for letting the user see a pre-prepared list of items. For example, the program can inform the user that the items in the list box are the list of all the presidents of the USA, a list of fonts, a list of phone numbers and so on.

It would be nice if the user could "pick" up an item from the list and "do something" with the picked item. In the next sections you'll now implement a mechanism that lets the user pick up an item from the list.

Placing an Edit Box Control inside Form1

You'll now place a Text Box inside Form1.

 Place a Text Box inside Form1. Then set the properties of the textbook as follows:




Name: Text1


Text: Make it empty

The reason you were instructed to place a Text Box is that in the following sections you'll be instructed to add code that lets the user pick up an item from the list box and place the item inside the Text Box.

Implementing an Edit Menu

You'll now implement an **Edit** menu. The **Edit** menu will have the following menu items in it:

-  Copy
-  Paste
-  Cut

 Make sure that Form1 is the selected window, then select **Menu Editor** from the **Tools** menu of Visual Basic.

 Implement the following menu:

Caption	Name	Shortcut
&Edit	mnuEdit	
...&Copy	mnuCopy	Ctrl-C
...&Paste	mnuPaste	Ctrl-V
...Cu&t	mnuCut	Ctrl-X

The **Edit** menu should look as shown in Figure 25.5.

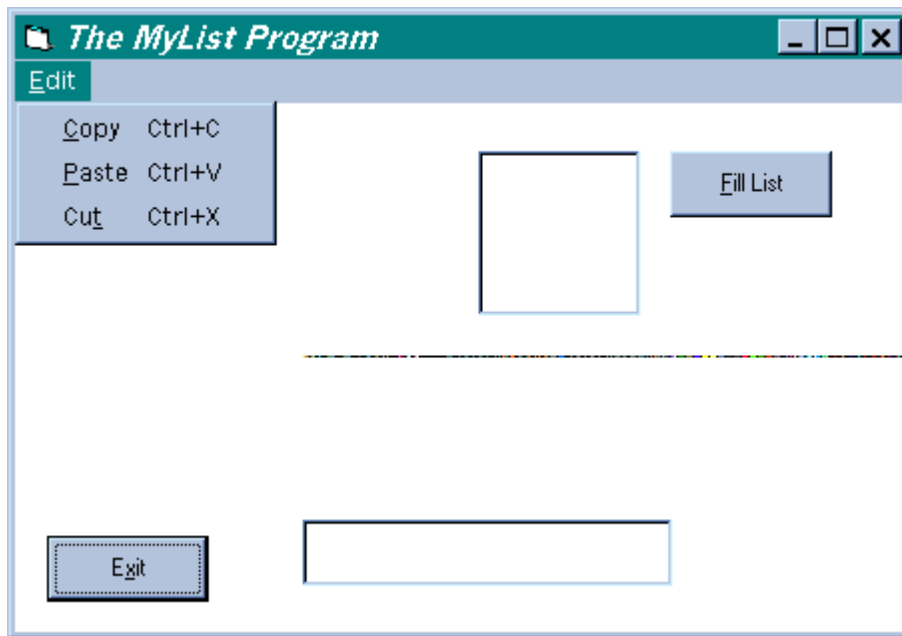


Figure 25.5. The Edit menu of the MyList program

An Active Control (Control with a Focus)


Before going on with the implementation of the MyList program, let's demonstrate the meaning of the term **Active Control**. This term is also referred to as the **Control that has the Keyboard Focus**, or the **Control with the Focus** for short.

To better understand what is the currently active control, perform the following experiment:


 Execute the MyList program.


 Click inside the text box.

As a result, a blinking cursor appears inside the text box. The blinking cursor is an indication that the user can now use the keyboard, and as a result, the text that is typed will be displayed inside the text box. Putting it in other words, the text box has the keyboard focus because the output of the keyboard (the typed characters) are directed to the text box.


 Press the Tab key of your keyboard several times.

As you press the Tab key, notice that the keyboard focus is shifted to other controls. Each control has its own visual indication that the control has the keyboard focus. When the CommandButton has the keyboard focus, a dashed rectangle surrounds the caption of the button.

 Make the **Fill List** button the active control (that is, press the Tab key until the **Fill List** button has a dashed rectangle around the caption of the **Fill List** button.)


 Now that the **Fill List** button has the keyboard focus, you can use the keyboard to send commands to the button. Press the Space bar key on your keyboard.

The result is that the list box is filled. In other words, pressing the space bar key while the **Fill List** button has the keyboard focus, has the same effect as clicking the **Fill List** button.

 Experiment with the MyList program. Press the Tab key on your keyboard several times, and note the visual indication that the control gives whenever the control has the focus.


Note that there is an alternative way to make the control the active control. If you click the control, you make it the active control. For example, make the text box the active control, then click the **Fill List**

button, and notice that the focus is shifted from the text box to the **Fill List** button.

 Click the **Exit** button to terminate the program.

Implementing the Copy Mechanism

You'll now implement the copy mechanism. That is, you'll write code that copies text from the list box to the Clipboard of Windows.

 Type the following code inside the **mnuCopy_Click()** procedure:

```
Private Sub mnuCopy_Click()  
  
Clipboard.Clear  
  
If TypeOf Screen.ActiveControl Is ListBox Then  
  
    Clipboard.SetText Screen.ActiveControl.Text  
  
End If  
  
End Sub
```

Let's go over the code that you typed inside the **mnuCopy_Click()** procedure:

You cleared the contents of the Clipboard with the **Clear** method:

```
Clipboard.Clear
```

Note that in the preceding statement you applied the **Clear** method on the **Clipboard** object. The **Clipboard** object is the clipboard of Windows. Every Windows application has a "right" to copy data into the clipboard and copy data from the clipboard.

=====

The next statements that you typed are as follows:

```
If TypeOf Screen.ActiveControl Is ListBox Then  
  
    Clipboard.SetText Screen.ActiveControl.Text  
  
End If
```

The **If** condition checks if the control with the focus is the list box. Note how the **If** condition is constructed:

```
If TypeOf Screen.ActiveControl Is ListBox Then
```

The words **Screen.ActiveControl** mean the control which currently has the keyboard focus. The word **TypeOf** means the type of control. Possible values for the **TypeOf** are for example **ListBox**, **TextBox**, **ComboBox**, and **PictureBox**. Putting it altogether, the preceding **If** statement checks if the control that has the keyboard focus is a **ListBox** control.

The code that you typed under the **If** statement is therefore executed whenever the **List1** list box has the keyboard focus. Here is the code under the **If** statement:

```
Clipboard.SetText Screen.ActiveControl.Text
```

The preceding statement uses the **SetText** method on the **Clipboard** object. **SetText** places text inside the Clipboard.

You supplied as the parameter of the **SetText** method the following value:

Screen.ActiveControl.Text


Screen.ActiveControl.Text represents the text which is currently highlighted inside the list box.

=====


Putting it all together, the code under the **If** statement copies the highlighted item of the list box into the Clipboard.


Before going on with the development of the program, lets see the code that you typed so far in action:


 Execute the MyList program.

 Click the **Fill List** button to fill the list box with items. Highlight an item inside the list box, and press Ctrl+C (or select **Copy** from the **Edit** menu).


The MyList program responds by copying the highlighted text to the Clipboard.

 Start a program such as Notepad, Word for windows, or any other text editor or word processor program.

 Place the mouse cursor inside the document of the text editor or word processor program that you started in the previous step.

 Paste the Clipboard contents into the document. (Almost all text editor and words processors programs let you paste the Clipboard contents by either selecting the **Paste** menu item from the **Edit** menu of the program, or pressing **Ctrl+V** on the keyboard).


As you can see, the clipboard contents and been pasted to the document. You successfully copied the highlighted item of the list box to the clipboard of Windows, and then you paste the clipboard contents into the document that you started in your text editor or Word processor.

 Experiment with the MyList program, and then click the Exit button to terminate the MyList program.

=====

Implementing the Paste Mechanism

The **Copy** mechanism that you implemented copy the highlighted item in the List box to the Clipboard. You'll now implement the **Paste** mechanism so that the user will be able to paste the contents of the Clipboard to the text box of the MyList program.

 Add the following code inside the **mnuPaste_Click()** procedure:

```
Private Sub mnuPaste_Click()
```

```
If TypeOf Screen.ActiveControl Is TextBox Then  
    Screen.ActiveControl.SelText = Clipboard.GetText()  
End If
```

```
End Sub
```

The code that you typed uses an **If** statement to examine if the active control (e.g., the control with the keyboard focus) is the text box. If so, the code under the **If** statement is executed.


The code under the **If** statement copies the contents of the Clipboard to the text box:

```
Screen.ActiveControl.SelText = Clipboard.GetText()
```

The preceding statement gets the contents of the Clipboard, and this contents is copied into the active control (which in this case is the text box).


Let's see your code in action:

 Execute the MyList program.


 Click the **Fill List** button to fill the list box with items. Highlight an item inside the list box, and then press **Ctrl+C** (or select **Copy** from the **Edit** menu).

The MyList program responds by copying the highlighted text to the Clipboard.

 Click inside the text box to make the text box the active control.


 Paste the contents of the Clipboard into the text box by pressing **Ctrl+V** (or by selecting **Paste** from the **Edit** menu).


The MyList program responds by pasting the contents of the Clipboard into the text box.


 Experiment with the MyList program, then click the **Exit** button to terminate the program.

NOTE


Just as you were able to copy an item from the list box into the Clipboard and then paste the contents of the clipboard into a document of another text editor/Word processor program, you can also do the following:

 Open a new document with your text editor or Word processor program.

 Type something inside the document, highlight the text the you typed. And then press **Ctrl+C** (or select **Copy** from the **Edit** menu of your Text editor/Word processor program).

 Switch back to the MyList program, click inside the text box, and then press **Ctrl+V** (or select **Paste** from the **Edit** menu).

The MyList program responds by pasting the contents of the clipboard to the text box.

 Click the Exit button to terminate the MyList program.

Copying Text From the Text Box to the List Box

So far, you implement code that lets the user copy text from the list box to the text box. You'll now write code that lets the user copy text from the text box to the list box.

 Modify the code inside the **mnuCopy_Click()** procedure so that it will look as follows:

```
Private Sub mnuCopy_Click()  
  
Clipboard.Clear  
  
If TypeOf Screen.ActiveControl Is ListBox Then  
  
    Clipboard.SetText Screen.ActiveControl.Text  
  
ElseIf TypeOf Screen.ActiveControl Is TextBox Then  
  
    Clipboard.SetText Screen.ActiveControl.SelText  
  
End If  
  
End Sub
```

You added an **Elseif** statement:

```
ElseIf TypeOf Screen.ActiveControl Is TextBox Then
```

```
Clipboard.SetText Screen.ActiveControl.SelText
```

The preceding **Elseif** statement checks to see if the active control is the text box. If so, the code under the **Elseif** statement is executed. This code copies the highlighted text in the text box to the Clipboard.

 Modify the code inside the **mnuPaste_Click()** procedure so that it will look as follows:

```
Private Sub mnuPaste_Click()
```

```
If TypeOf Screen.ActiveControl Is TextBox Then
```

```
Screen.ActiveControl.SelText = Clipboard.GetText()
```

```
ElseIf TypeOf Screen.ActiveControl Is ListBox Then
```

```
Screen.ActiveControl.AddItem Clipboard.GetText()
```

```
End If
```

```
End Sub
```

You added an **Elseif** statement as follows:

```
ElseIf TypeOf Screen.ActiveControl Is ListBox Then
```

```
Screen.ActiveControl.AddItem Clipboard.GetText()
```

The **Elseif** statement that you added checks if the active control is the list box. If so, the contents of the Clipboard is copied to the list box as follows:

=====

```
Screen.ActiveControl.AddItem Clipboard.GetText()
```

So the text that you copy to the list box is added as a new item to the list box.

Let's see your code in action:


 Select **Save Project** from the **File** menu of Visual Basic to save your work.

 Execute the MyList program.


 Click inside the text box, and type something in it.

 Highlight the text that you typed inside the text box, and then press **Ctrl+C** on your keyboard (or select **Copy** from the **Edit** menu).

The MyList program responds by copying the highlighted text into the Clipboard.

 Click inside the List box, and then press **Ctrl+V** on your keyboard (or select **Paste** from the **Edit** menu).

The MyList program responds by adding the Clipboard contents as a new item to the list. The item is added as the last item in the list (so you'll have to use the scrollbar of the list box to see the last item that you added to the list box).

 Experiment with the MyList program, then click the **Exit** button to terminate the program.

Implementing the Cut Mechanism

=====

You'll now implement the **Cut** mechanism of the MyList program. The **Cut** feature is the process of highlighting a text, and then performing two operations:

- Copying the highlighted text to the Clipboard.
- Deleting the highlighted text.

 Type the following code inside the **mnuCut_Click()** procedure:

```
Private Sub mnuCut_Click()  
  
If TypeOf Screen.ActiveControl Is ListBox Then  
  
    If Screen.ActiveControl.ListIndex >= 0 Then  
  
        Screen.ActiveControl.RemoveItem _  
            Screen.ActiveControl.ListIndex  
  
    End If  
End If  
  
End Sub
```

The code that you typed uses an **If** statement to determine if the list box is the active control:

```
If TypeOf Screen.ActiveControl Is ListBox Then  
    ...  
    ...  
    ...  
End If
```

If the list box is the active control, an inner **If** statement is executed:

=====

```
If Screen.ActiveControl.ListIndex >= 0 Then

    Screen.ActiveControl.RemoveItem _
        Screen.ActiveControl.ListIndex

EndIf
```

The inner **If** statement checks to see if there are items in the list box. You need to check if there are items in the list box, because you are about to delete an item from the list, and therefore you must first verify that there is an item to delete. **ListIndex** is the number of the item in the list that is currently highlighted. The first item has an index number **0**, the second item in the list has the index number **1**, and so on. If the list in the list box is empty, the list box can still be the active control, but the **ListIndex** property will be a number less than **0**. Thus, if the condition of the inner **If** statement is satisfied, it means that there is at least one item in the list.

Here is the code under the inner **If** statement:

```
Screen.ActiveControl.RemoveItem _
    Screen.ActiveControl.ListIndex
```

The preceding code uses the **RemoveItem** method to remove the item that is currently highlighted. The currently highlighted item is:


```
Screen.ActiveControl.ListIndex
```

Before continuing with the implementation of the **Cut** feature, let's see what you accomplished so far:


 Select **Save Project** from the **File** menu of Visual Basic to save your work.

 Execute the MyList program.

 Click the **Fill List** button to fill the list box with items.

 Click one of the items in the list, and then press **Ctrl+X** on your keyboard (or select **Cut** from the **Edit** menu).

The MyList program responds by deleting the highlighted item from the list.

 Experiment with the MyList program by deleting all the items from the list, then click the Exit button to terminate the MyList program.

As stated, the **Cut** mechanism involves two steps: copying the highlighted text to the Clipboard and then deleting the highlighted text. So far you implemented only the deleting portion of the **Cut** feature.

 Modify the **mnuCut_Click()** procedure so that it will look as follows:

```
Private Sub mnuCut_Click()
```

```
mnuCopy_Click
```

```
If TypeOf Screen.ActiveControl Is ListBox Then
```

```
...
```

```
...
```

```
...
```

```
End If
```

```
End Sub
```

You execute the **mnuCopy_Click** procedure before deleting the highlighted text:

```
mnuCopy_Click
```


```
=====
```

Let's see your code inaction:


 Select **Save Project** from the **File** menu of Visual Basic to save your work.

 Execute the MyList program.


 Click the **Fill List** button to fill the list box with items.

 Click one of the items in the list, and then press **Ctrl+X** on your keyboard (or select **Cut** from the **Edit** menu).

The MyList program responds by deleting the highlighted item from the list. Also, the deleted item is copied to the clipboard.

 Click inside the text box, and then press **Ctrl+V** on your keyboard (or select **Paste** from the **Edit** menu).

The MyList program responds by copying the contents of the clipboard to the text box. As you can see, the text that was copied to the text box is the item the you deleted from the list box.

 Experiment with the MyList program, then click the **Exit** button to terminate the program.

Cutting Text from the Text Box

You'll now add code that performs the **Cut** operation on an highlighted text inside the text box:

 Modify the **mnuCut_Click()** procedure so that it will look as follows:

```
Private Sub mnuCut_Click()
```

=====

```
mnuCopy_Click
```

```
If TypeOf Screen.ActiveControl Is ListBox Then
```

```
    If Screen.ActiveControl.ListIndex >= 0 Then
```

```
        Screen.ActiveControl.RemoveItem _  
            Screen.ActiveControl.ListIndex
```

```
    End If
```

```
ElseIf TypeOf Screen.ActiveControl Is TextBox Then
```

```
    Screen.ActiveControl.SelText = ""
```

```
End If
```

```
End Sub
```

You added an **Elseif** statement that checks if the active control is the text box, and if so, the highlighted text is replaced with null:


```
ElseIf TypeOf Screen.ActiveControl Is TextBox Then
```

```
    Screen.ActiveControl.SelText = ""
```


Let's see your code in action:




Select **Save Project** from the **File** menu of Visual Basic to save your work.

 Execute the MyList program. Type something inside the text box, highlight the text, and then press **Ctrl+X** on your keyboard (or select **Cut** from the **Edit** menu).

The MyList program responds by deleting the highlighted text (and as you know, the deleted highlighted text was copied into the Clipboard).

 Click inside the list box, and then press **Ctrl+V** on your keyboard (or select **Paste** from the **Edit** menu).

The MyList program responds by adding an item to the list. Scroll down the list, and verify that the added item is the text that you deleted from the text box.

 Experiment with the MyList program, and then click the Exit button to terminate the program.

What You Accomplished in This Lesson



You completed Lesson 25 of the Self Study Visual Basic tutorial. In this lesson you learned to write programs that utilize list boxes and text boxes. In this lesson you also learned how to perform the standard **Copy**, **Paste**, and **Cut** clipboard operations.

Frequently Asked Questions



Q1. In the MyList program, I implemented the **Ctrl+X** keys as a short cut for selecting **Cut** from the **Edit** menu. I don't see the connection between **Ctrl+X** and **Cut**. Maybe **Ctrl+T** is a better choice for this short cut?

A1. User **Ctrl+X** as a short cut for **Cut**, use **Ctrl+C** as a short cut for **Copy**, and use **Ctrl+V** as a short cut for **Paste**. These shortcuts are used for many years (since the time of DOS). People expect your program to use these short cuts. If you use anything else, you'll probably annoy your users!

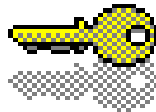
Exam



1. What does the following statement check?

```
If TypeOf Screen.ActiveControl Is TextBox Then
```

Answers to Exam

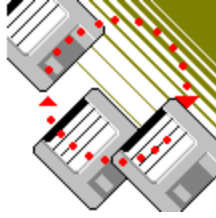


1. The statement


```
If TypeOf Screen.ActiveControl Is TextBox Then
```

checks if the active control is the text box control.


Project




Place another text box inside Form1. Then write code that performs **Copy**, **Paste**, and **Cut** operations on the new text box.

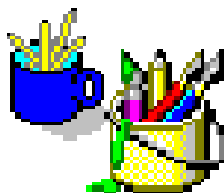
 Place another text box inside Form1.

That's it! You do **not** have to write any code! You already typed the code that performs the **Copy**, **Paste**, and **Cut** operations!


 Experiment with the MyList program and verify that you can perform all the Clipboard operations. Copy, paste and cut text from the list box into either of the text boxes, as well from either of these text boxes to the list box.

 Click the Exit button to terminate the program.


Cosmetic Considerations




For the purpose of learning the material, the list box was filled with items inside the **cmdFillList_Click()** procedure. Typically, you will fill the list automatically, without the need to click any button.

 Copy the code that resides inside the **cmdFillList_Click()** procedure into the **Form_Load()** procedure.

 Delete the **cmdFillList** button from Form1.

 Execute the MyList program, and notice that the list box is filled automatically upon starting the program.

 Experiment with the MyList program, then click the Exit button to terminate the program.

The **Str()** and **Format()** functions

You added items to the list box as follows:

```
For Counter = 0 To 8
    List1.AddItem Str(ItemsOfList(Counter))
Next
```

Note that the **Str()** function is used for converting numbers to strings. When the number **100** is converted to a string with the **Str()** function, the following string is generated: " 100"

In other words, a space is inserted as the first character in the string. To avoid generating the space character, use the **Format()** function instead of the **Str()** function as follows:

 Modify the code inside the **Form_Load()** procedure so that it will look as follows:

```
Private Sub Form_Load()
```

```
Dim ItemsOfList(0 To 8)
```

=====

```
Dim Counter
```


```
ItemsOfList(0) = 100  
ItemsOfList(1) = 101  
ItemsOfList(2) = 102  
ItemsOfList(3) = 103  
ItemsOfList(4) = 104  
ItemsOfList(5) = 105  
ItemsOfList(6) = 106  
ItemsOfList(7) = 107  
ItemsOfList(8) = 108
```

```
For Counter = 0 To 8  
    List1.AddItem Format(ItemsOfList(Counter))  
Next
```

```
End Sub
```

So now you fill the items of the list box as follows:


```
For Counter = 0 To 8  
    List1.AddItem Format(ItemsOfList(Counter))  
Next
```


 Execute the MyList program, click the **Fill List** button, and verify that the items in the list box do not have a space as the first character of the items.

Sorting


When you add a new item to the list of the list box, the item is added as the last item in the list. Typically, you want the list to be sorted alphabetically. Here is how you do this:

 At design time, set the **Sort** property of the list box to **True**.

 Execute the MyList program, click the **Fill List** button to fill the list.

 Type **1** inside the text box, and then copy this text and paste it to the list box.


Because the list box is now sorted, **1** will appear as the first item.

 Experiment with the MyList program and then click its **Exit** button to terminate the program.

How To Contact TegoSoft



You can contact TegoSoft Inc. by any one of the following methods:

 Use TegoSoft Internet Web site:

<http://www.tegosoft.com>

 Send TegoSoft an e-mail:

tegosoft@msn.com

 Send TegoSoft a letter:

TegoSoft Inc.

P.O.Box 389

Bellmore, NY 11710


USA

Technical Support



If you have a technical question, you can post the question to the TegoSoft Technical Support staff, and they will try to answer your question.

The **best** way to post a technical question is by sending TegoSoft an e-mail.

 The e-mail of TegoSoft is:
tegosoft@msn.com

When sending TegoSoft an e-mail with a technical question, please follow the following format:

Date: _____
Your name: _____
Company (if applicable): _____
Your phone number: _____
Your e-mail: _____
Country (if not USA): _____
State (if inside USA): _____

Operating System used: _____
Programming language and version : _____

My technical question is:

Copyright © and Notices

Copyright © 1994, 1995, 1996 by TegoSoft Inc. ® All Rights Reserved



TegoSoft Self Study Tutorials & Software

Copyright ©1994, 1995, 1996 by TegoSoft Inc. ® All Rights Reserved

Although precaution was taken in the preparation of this document, TegoSoft assumes no responsibility for errors or omissions. TegoSoft is not liable for damages resulting from the use of the information contained in this document. Use this document at your own risk. This document is copyright protected. This means that you should treat this document like any other copyright material. No part of this document should be copied in any way. You are not allowed to use this document or any part of this document for any purpose other than read it. You are not allowed to publish this document or any part of this document in any way. You are not allowed to sell this document or any part of this document, you are not allowed to incorporate this document or any part of this document in any book, magazine, Web Site, Electronic forums, disks, CDs, or any other media. The only thing that you are allowed to do with this document is read it for the sole purpose of studying the material that is presented in this document. If this document includes software, the software must be treated in the exact same way that this document is treated. You are not allowed to distribute the software in any way. The sole purpose of supplying the accompanying software is to enable you to experience with it. No part of this document should be modified. If accompanying software is included with this document, you are not allowed to modify the software.

Rev. 031796-1