

Lab 5: Conditional Logic

Decision making process is an important part of programming in Visual Basic.NET because it can solve practical problems intelligently and provide useful output or feedback to the user. For example, we can write a Visual Basic.NET program that ask the computer to perform certain task when a certain condition is met, or a program that will reject non-numeric data. In order to control the program flow and to make decisions, we need to use the conditional operators and the logical operators together with the If control structure.

1. Conditional Operators

The conditional operators are powerful tools that resemble mathematical operators. These operators allow a Visual Basic.NET program to compare data values and then decide what actions to take, whether to execute a program or terminate the program and more. They are also known as numerical comparison operators. Normally they are used to compare two values to see whether they are equal or one value is greater or less than the other value. The comparison will return a true or false result. These operators are shown in the Table below.

Operator	Meaning
=	Equal to
>	More than
<	Less than
>=	More than or equal
<=	Less than or equal
<>	Not equal to

2. Logical Operators

Sometimes we might need to make more than one comparison before a decision can be made and an action taken. In this case, using numerical comparison operators alone is not sufficient, we need to use additional operators, and they are the logical operators. These logical operators are shown in the Table below.

Normally the conditional and logical operators are used to compare numerical data. However, you can also compare strings with the above operators. In making strings comparison, there are certain rules to follows: Upper case letters are less than lowercase letters, "A"<"B"<"C"<"D".....<"Z" and number are less than letters.

Operator	Meaning
And	Both sides must be true
Or	One side or other must be true
Not	Negates truth

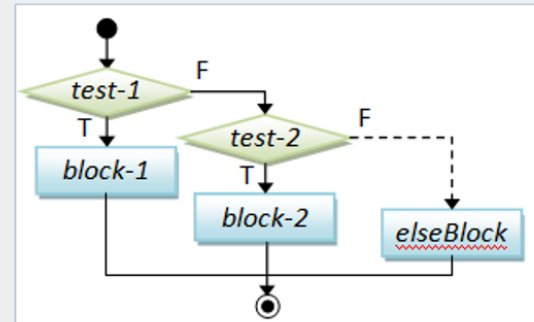
3. The If control structure

To effectively control the Visual Basic.NET program flow, we shall use the If control structure together with the conditional operators and logical operators. There are basically three types of If control structures, namely **If....Then** statement, **If....Then... Else** statement and **If....Then....Elseif** statement.

Syntax	Example	Flowchart
<pre> ' if-then If (booleanExpression) Then true-block End If </pre>	<pre> If (mark >= 50) Then Label1.Text = "Congratulation!" End If </pre>	<pre> graph TD Start(()) --> Test{test} Test -- T --> TrueBlock[true-block] Test -- F --> Join(()) TrueBlock --> Join Join --> End((())) </pre>
<pre> ' if-then-else If (booleanExpression) Then true-block Else false-block End If </pre>	<pre> If (mark >= 50) Then Label1.Text = "Congratulation!" Else Label1.Text = "Try Harder!" End If </pre>	<pre> graph TD Start(()) --> Test{test} Test -- T --> TrueBlock[true-block] Test -- F --> FalseBlock[false-block] TrueBlock --> End((())) FalseBlock --> End </pre>

' nested-if

If (booleanExpr-1) Then	If (mark >= 80) Then
block-1	Label1.Text = "A"
Elseif (booleanExpr-2) Then	Elseif (mark >= 70) Then
block-2	Label1.Text = "B"
Elseif (booleanExpr-3) Then	Elseif (mark >= 60) Then
block-3	Label1.Text = "C"
Elseif (booleanExpr-4) Then	Elseif (mark >= 50) Then
.....	Label1.Text = "D"
Else	Else
elseBlock	Label1.Text = "F"
End If	End If

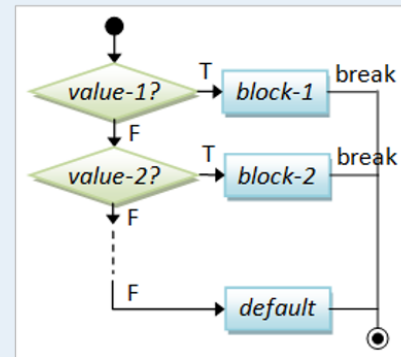


4. The Select Case control structure

The Select Case control structure is slightly different from the If....Elseif control structure. The difference is that the Select Case control structure basically only make decision on one expression or dimension (for example the examination grade) while the If ...Elseif statement control structure may evaluate only one expression, each If....Elseif statement may also compute entirely different dimensions. Select Case is preferred when there exist multiple conditions as using If...Then..Elseif statements will become too messy.

' select-case-default

Select Case (selector)	Select Case mark
Case value-1	Case Is >= 80
block-1	Label1.Text = "A"
Case value-2	Case Is >= 70
block-2	Label1.Text = "B"
Case value-3	Case Is >= 60
block-3	Label1.Text = "C"
.....	Case Is >= 50
Case value-n	Label1.Text = "D"
Case Else	Case Else
default-block	Label1.Text = "F"
End Select	End Select



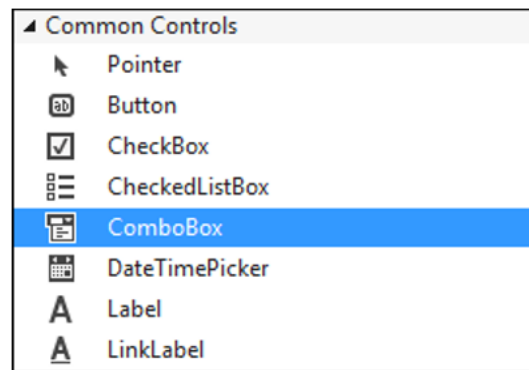
You can use Select Case statement with numbers, as well, and the To word comes in very handy here. If you were checking a variable to see if the number that was in the variable fell within a certain age-range, you could use something like this:

```
Select Case age_range
    Case 16 To 21
        MsgBox("Still Young")
    Case 50 To 64
        MsgBox("Start Lying")
End Select
```

Here, a variable called age_range is being tested. It's being checked to see if it falls between certain values. If it does, then a message box is displayed.

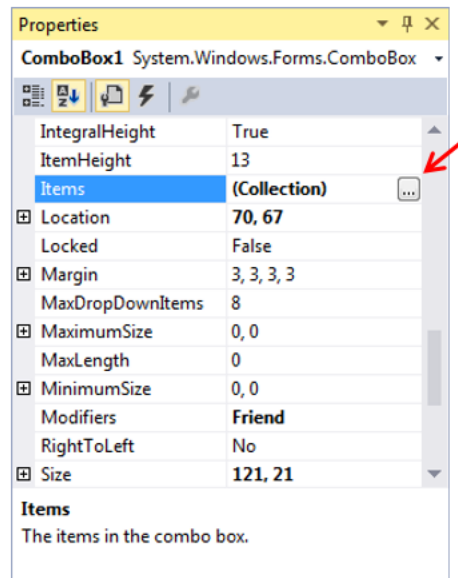
5. Add a Combo Box to a VB .NET form

Create a new project for this section. Add a button to your new form. Then, locate the Combo Box on the Visual Basic .NET toolbox. It looks like this:

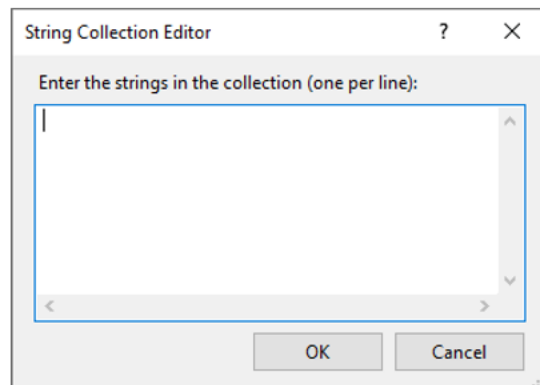


Double click the icon to add a Combo Box to your form. Or click once with the left hand mouse button, and then draw one on the form. A combo box is a way to limit the choices your user will have. When a black down-pointing arrow is clicked, a drop down list of items appears. The user can then select one of these options. With the Combo Box Controls displayed, do the following:

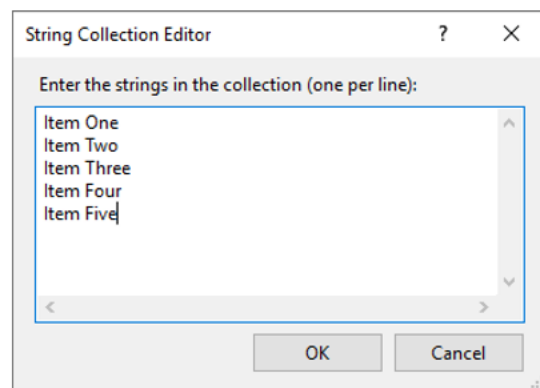
- Click on your Combo Box to select it. Then locate the Items property from the Properties window.



- Click the grey button, as above. The one with the three dots in it. When you do, you'll get the following box popping up:

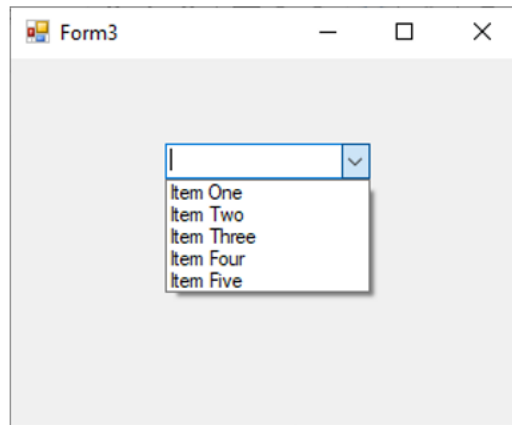


- To use the String Collection Editor, type an item and press Return (it's just like a normal textbox. Each item will be one item in your drop-down box.)
- Enter five items, as in the image below:



- Then click the OK button at the bottom.

The Editor will close, and it will look like nothing has happened. However, run your program and test out your new Combo Box. You should have something like this:



You now need to know how to get values from the list. Once you know how to get a value from the list, you can put the value into a variable and test it with some Conditional logic.

Getting a value from a Combo Box is fairly straightforward, because it acts just like a Textbox. A Textbox has a Text property, and so does a Combo Box. To get a value from a Textbox, you would code like this

```
MyVariable = Textbox1.Text
```

Whatever is in the Textbox will be transferred to the variable called MyVariable. The process is exactly the same for a Combo Box. The code is like this:

```
MyVariable = Combobox1.Text
```

Now we are transferring whatever is selected from the Combo Box to the variable called MyVariable.

Let's try it. Double click the button you added to your form. This will open the code window. Then enter the following code for the button:

```
Dim MyVariable as String  
MyVariable = Combobox1.Text  
MsgBox(MyVariable)
```

Run your program. When the program is running, select an item from your Combo Box. Then click your button. Whatever was in the Combo Box window should have ended up in the Message Box.

Exercise 1

Start a new project. Add a textbox and a button to your new Form. Then write a program that does the following:

1. Asks users to enter a number between 10 and 20.
2. The number will be entered into the Textbox.
3. When the Button is clicked, your Visual Basic code will check the number entered in the Textbox.
4. If it is between 10 and 20, then a message will be displayed.
5. The message box will display the number from the Textbox.
6. If the number entered is not between 10 and 20 then the user will be invited to try again, and whatever was entered in the Textbox will be erased

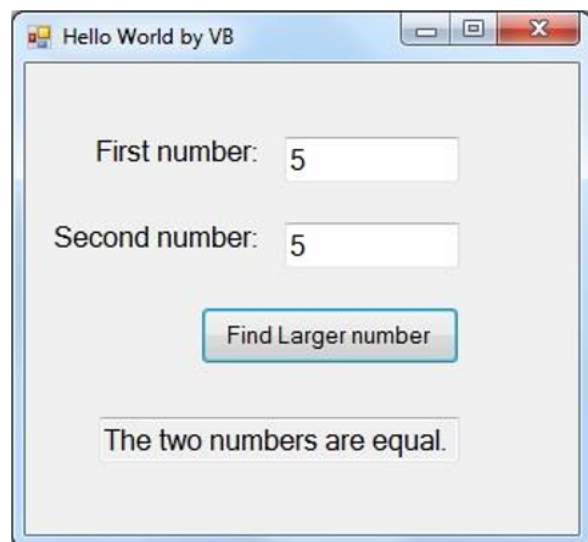

Exercise 2

Add a Combo box, a new label and another button to your form. Create a list of items for your Combo Box. The list of items in your Combo box contains four options like: Sunny, Snowy, Rainy, and Cloudy. Then try the following:

Use a select case statement to test what a user has chosen from your drop-down list. Display an appropriate message when the button was clicked. The message should give the weather report in words. For example, if the user chooses the Sunny, you might **display “It looks like sunny weather today”**.

Exercise 3

Create a program to compare the two numbers input by the user. Use a textbox to display the largest number or report if the two numbers are equal.

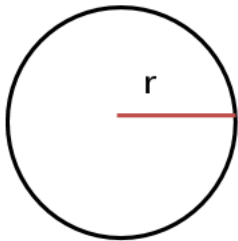


Exercise 4

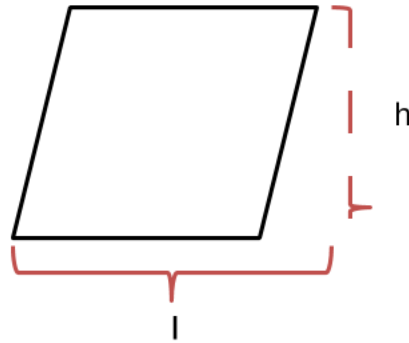


Genius question

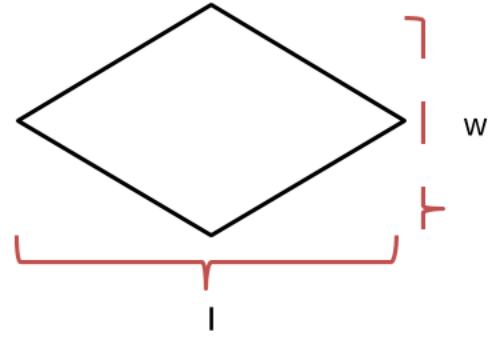
The figure below shows some geometric shapes and formulas for their areas. Write a program that requests the user to select one of the shapes, request the appropriate lengths, and then gives the area of the figure.



Circle
 $3.141593 * r^2$



Parallelogram
 $l * h$



Kite
 $(l * w) / 2$