Learning VB.Net

Tutorial 05 – understanding variables

Hello everyone… welcome to vb.net tutorials. These are going to be very basic tutorials about using the language to create simple applications, hope you enjoy it. If you have any notes about it, please send them to notes@mka-soft.com I will be happy to receive them. Finally if you find these tutorials are useful, it would be nice from you to send a small donation via PayPal to donation@mka-soft.com.

tutorial posted on 2009-June-18.

Hello everyone. Today's tutorial is about variables. When a computer processes information or data, that data should be placed in its memory first, then it performs different operations on that. You application as well should process the information in memory. To accomplish that the you allocate memory by defining variables. Simply, a variable is small piece of memory that allows the program to process data within it.

To define a variable in VB.NET you use the following format:

```
        Dim MyInt As Integer
```

Where the `Dim` tells the computer that you are going to define a variable. Next you write variable name (in this case it is `MyInt`) and finally the data type `As Integer` which tells the computer you are going to perform integer operations on the data in that variables.

So, in order to try out integer variables, create a simple windows application, and place a button on your form, and modify its event so that it looks something similar to this:

```
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
                Handles Button1.Click
        Dim MyInt As Integer
        MyInt = 55 + 4
        MsgBox(MyInt)
    End Sub
```

Try this code out… you see that MyInt stores the result of the operation. You can place any type of operation like +,-,*,/ and more… but for now let us try to create another variable and use it with this one. Modify the code to be something like this:

```
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
                Handles Button1.Click
        Dim MyInt As Integer
        Dim A As Integer, B As Integer
        A = 10
        B = 30
        MyInt = A + B + 2
        MsgBox(MyInt)
    End Sub
```

In this case we are using 3 variables (A,B,MyInt). We compute using the values inside the A,B and store the result in MyInt. So basically any of the following statements is valid:

```
        MyInt = 22                            places value 22 in MyInt
        MyInt = A                             copies the value of A into MyInt (10)
        MyInt = B                             copies the value of B into MyInt (30)
        MyInt = A + B                         adds A,B and places the result in MyInt (40)
        MyInt = A - B                         subtructs B from A (-20)
        MyInt = A + A + B + 2                 adds double value of A, value of B and 2 to
                                              the total (52)
        MyInt = B*A-A                         multiply A by B then subtructs A (290)
        MyInt = A/2                           divides A by 2 (10)
        MyInt = B/A                           divides B by A (3)
        MyInt = MyInt + 1                     gets the value of MyInt (usually 0) and adds
                                              one to that (total is 1)
```

and you can come up with any form of statement that helps you solve your problem.

Now we start understanding what is the data type. Each variable can be one data type. The data type tells the computer what you are going to process in this variable. For example **Integer** means you are working with whole numbers (like 30,40,55) and not fractions. So if you try out:

```
MyInt = 3.4
```

What you get is the value 3. That is because this data type can not store floating point numbers. To solve this issue you should use **Singe**. To make things clear try to execute the following code:

```
Dim I As Integer
I = 22 / 7
MsgBox("the PI (int) :" & I)

Dim S As Single
S = 22 / 7
MsgBox("the PI (single) :" & S)
```

When you run this code, you will see that it give you 3 as the value of PI (the integer case), then it gives you a value of 3.142857 for PI (the single case).

So why do you use integers and singles? Why not use single all the time? The answer is performance and storage. The computer processes integer operations faster than that of floating point numbers. And also some data types takes less storage than others. Because of that if you are going to develop an application that performs lots integer operations it makes sense to use integers to speed things up. In the end you select variable types depending on the nature of your problem.

There are similar data types to Integer and Single. These are Long (to store integers) and Double(to store floating point numbers). The difference between these and the previous ones is that they can represent wider range of numbers. To demonstrate this try out the following code:

```
MsgBox("integer " & Integer.MaxValue)
MsgBox("long " & Long.MaxValue)
MsgBox("single " & Single.MaxValue)
MsgBox("double " & Double.MaxValue)
```

The `Integer.MaxValue` gets the maximum value that an Integer can store. The same is true for other data types. Of course better representation requires more memory and also more processing time. Another example to demonstrate this is by calculating the value of PI using Double and Single. Try out the following code:

```
Dim I As Integer
I = 22 / 7
MsgBox("the PI (int) :" & I)

Dim S As Single
S = 22 / 7
MsgBox("the PI (single) :" & S)

Dim D As Double
D = 22 / 7
MsgBox("the PI (double) :" & D)
```

I did not write the Long data type here, but you can try it. It gives the same result as that of the integer case because Long data type can only store non-floating point values.

Another type of variables is the ones used to store complete statements, the String data type.

```
Dim Str1 As String
Dim Str2 As String

Str1 = "hello"
Str2 = " my friend"

MsgBox(Str1)
MsgBox(Str2)
```

As you can see, this data type stores letters and symbols. And actually you can do a number of operations on these. The simplest is to use the & operator to combine two strings.

```
Dim Str1 As String
Dim Str2 As String
Dim Str3 As String

Str1 = "hello"
Str2 = " my friend"
Str3 = Str1 & Str2

MsgBox(Str3)
```

You can see that Str3 now holds the complete statement “hello my friend”. More into string data type will come with time, but here we are focused on the very basics of variables.

Another important data type is the Boolean data type. This one is used to evaluate logic operations and it can only store True and False.

```
Dim B1 As Boolean
Dim B2 As Boolean
Dim B3 As Boolean
Dim B4 As Boolean

B1 = True
B2 = False
B3 = 88 > 10
B4 = "asmith" > "john"
MsgBox("b1 is:" & B1)
MsgBox("b2 is:" & B2)
MsgBox("88 > 10 is: " & B3)
MsgBox("asmith comes after john is:" & B4)
```

As you can see above B3 will check if 8 is greater than 10, and if so, it stores the value True, otherwise it stores False. B4 here shows how you can compare two strings. It checks to see if **asmith** comes alphabetically after **john** (which is the meaning of > sign in string comparison), and obviously this is not correct, so the value of B4 is false.

Finally you have the Date data type which is used to store the time and date. You can test it using the example below:

```
Dim D1 As Date
Dim D2 As Date
Dim D3 As Date
Dim D4 As Date
D1 = Now
D2 = Now.Date
D3 = "8:10:22 AM"
D4 = "2009-03-01"
MsgBox(D1)
MsgBox(D2)
MsgBox(D3)
MsgBox(D4)
```

The **Now** in the above example is used to get current system date and time. **Now.Date** is used to get system date alone.

There is actually not much in this tutorial, it is just a basic introduction to variables. The next tutorial starts working with variables, where you can get deep and better understanding of the variables. You can check the code on the web site, it might be a little bit different. Also check out the video tutorials on you tube.

If you have any notes please email me at **notes@mka-soft.com**. I will be happy to listen to your suggestions.