Software Design and Development

3 Number Input-Output

Terry Marris July 2009

Previously we looked at text input-output. Now we look at number input-output.

3.1 Integers

An integer is a whole number. 2, 4, 6 and 0, -1, -3 are all integers. 3.1416 is not an integer because it is not a whole number - it contains a decimal point.

3.2 Variables

A variable is a location in a computer's memory. It has:

- a name, e.g. number
- a type, e.g. Integer and
- a value e.g. 5.

To define a variable named *intNumber* of type *Integer* in VB we write

```
Dim intNumber As Integer
```

Dim is a VB word that introduces a variable definition.

intNumber is the name of the variable.

As Integer defines its type. This means that *intNumber* can store only whole numbers, and nothing else.

At this point, our variable, *intNumber*, has no defined value.

We can write:

```
Dim intNumber As Integer
intNumber = 5
```

This gives our variable *intNumber* the value 5.

3.3 Number Input

The essence of number input is:

- read the text input by the user
- convert that text to an integer

But this is fraught with problems. Users are idiots. Instead of typing in a number when requested, they might type nothing or letters of the alphabet instead - this gives us an invalid format error. Or they might type in a number too large for the poor computer to store in its finite memory - this gives us an overflow error. Us programmers, the brains of society, are obliged to catch these errors and deal with them kindly. Luckily for us, VB2008 makes such error catching easy.

```
try
    read the text by the user
    convert that text to an integer
catch invalid format errors
    offer helpful error message
catch number too large errors
    offer helpful error message
endTry
```

Well, that's the idea. Here is some VB code:

```
Try
    Dim intNumber As Integer = Convert.ToInt32(txtNumber.Text)
    Dim intResult = 2 * intNumber
    txtResult.Text = intResult.ToString()
Catch ex As FormatException
    txtError.Text = "Error: not a number"
Catch ex As OverflowException
    txtError.Text = "Error: number too big"
End Try
```

Try is a VB word. It introduces the mechanism to catch errors.

Convert.ToInt32(txtNumber.Text) converts the text input by the user in the text box to an integer - if it can. This integer number is then stored in *intNumber*.

The = is the assignment operator. It copies from right to left.

*2 * intNumber* multiplies the value stored in *intNumber* by 2 - if it can. The answer is stored in *intResult.*

intResult.ToString() converts the integer to a string. A string is just a programmers word for text. The string is shown in the *txtResult* text box.

Catch is a VB word. It introduces the type of error to be caught. *ex* is a variable name chosen by the programmer; it stands for exception - the programmer's word for an error. *FormatException* is the name given by VB to errors caused by attempting to convert none-numeric text into a number.

txtErrorMessage.Text = "Error: not a number" is the helpful error message we give to the user.

An *OverflowException* is caused by attempting to store a value that is too large for the variable.

End Try just ends the *Try ... Catch ...* structure.

Sure, this might be a bit overwhelming to some beginning programmers - but after you have typed it in and used it a few times it becomes natural and easy.

3.4 Number Output

Number output is easy: we just convert the number to text with *ToString()* like this:

```
txtResult.Text = intResult.ToString()
```

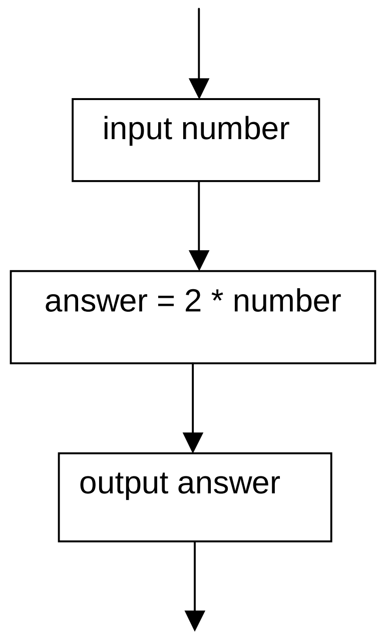
3.5 Number IO

Specification The specification of our example program is: read in a number input by the user, and output twice that number.

Design - structured English

```
read in number
answer = 2 * number
write out answer
```

Design - program flowchart



User Interface

1. Start a new project (chose **File**, **New Project)** and name it *NumberIO*

2. Set File Name = *NumberIO.vb*

3. Set Form
    a. Name = frmNumberIO
    b. Text = *Number Input Output*

4. Drag controls onto the form and set their properties as listed below:

    a. Label
        i. name = *lblNumber*
        ii. Text = *Number?*

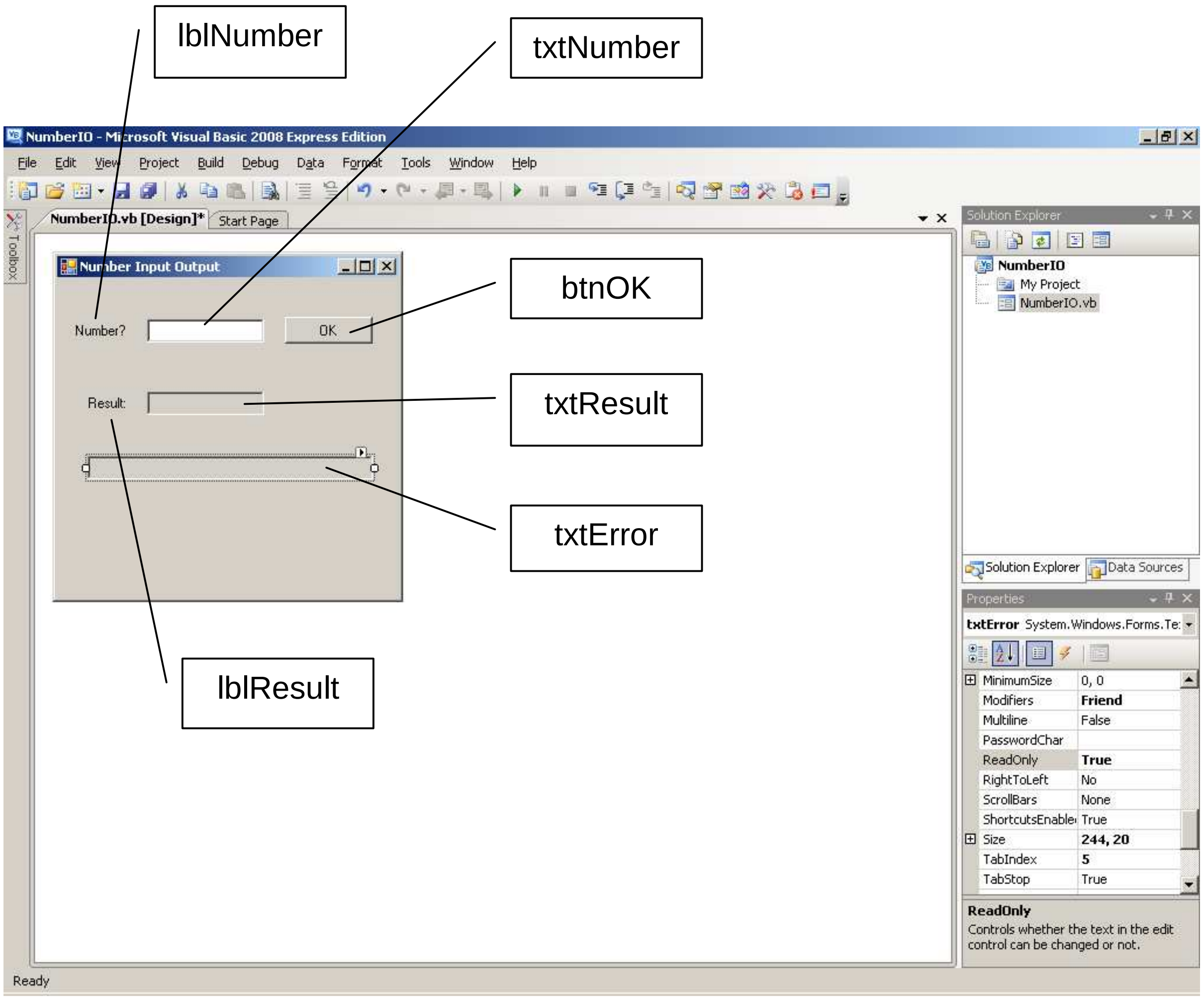
    b. Text Box
        i. name = *txtNumber*

    c. Button
        i. name = *btnOK*
        ii. Text = *OK*

    d. Label
        i. name = *lblResult*
        ii. Text = *Result:*

e. Text Box
   i. Name = *txtResult*
   ii. Read Only = *True*

f. Text Box
   i. Name = *txtError*
   ii. Read Only = *True*



txtError System.Windows.Forms.Te:

MinimumSize	0, 0
Modifiers	**Friend**
Multiline	False
PasswordChar	
ReadOnly	**True**
RightToLeft	No
ScrollBars	None
ShortcutsEnable	True
Size	**244, 20**
TabIndex	**5**
TabStop	True

ReadOnly
Controls whether the text in the edit control can be changed or not.

Programming Code

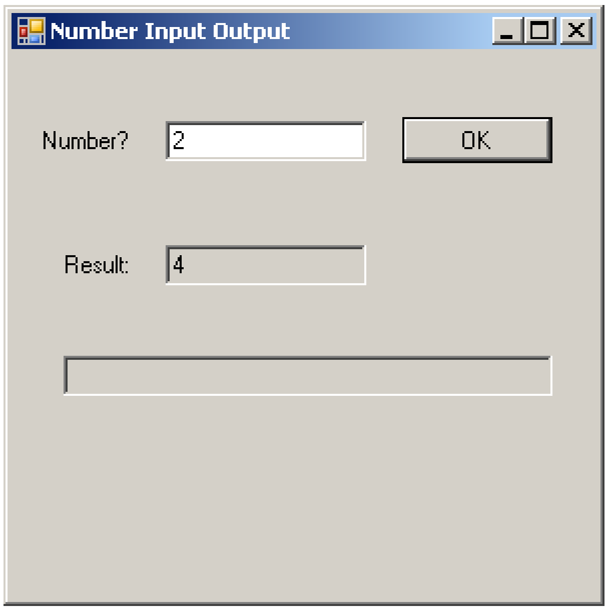
5. Write the VB code

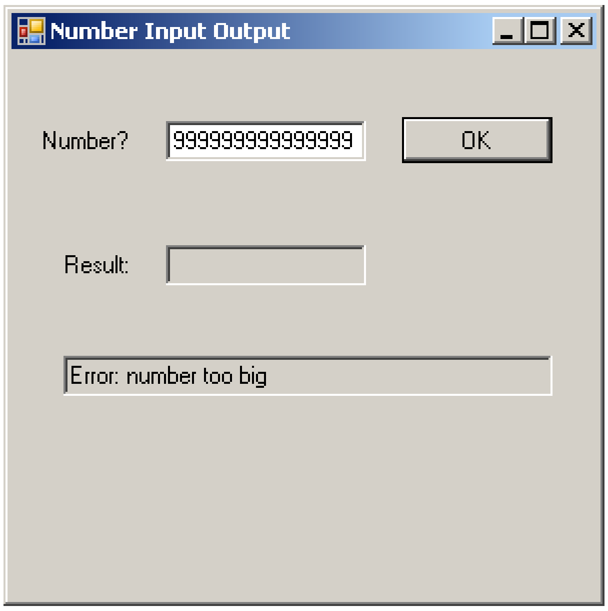
a. rapid double click on the OK button

b. between Private Sub btnOK_Click... and End Sub enter

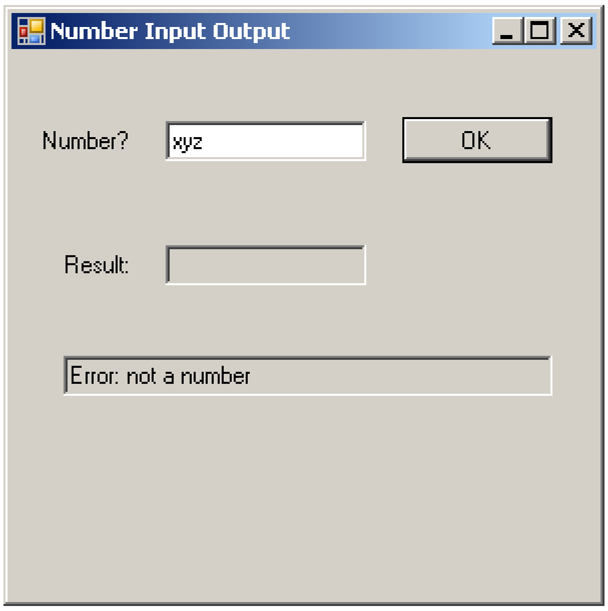
```
Try
  Dim intNumber As Integer = Convert.ToInt32(txtNumber.Text)
  Dim intResult = 2 * intNumber
  txtResult.Text = intResult.ToString()
Catch ex As FormatException
  txtError.Text = "Error: not a number"
Catch ex As OverflowException
  txtError.Text = "Error: number too big"
End Try
```

6. Run the program and check that your errors work as expected.





Error: number too big



Error: not a number

7. Print, and save your program in an appropriate place.

Exercise 3.1

1. Try out the example program, shown above, that inputs a number, multiplies it by 2, and then outputs the answer. Remember to include appropriate comments.

2. Write separate programs, one for each specification shown below. Remember to include appropriate comments and error traps, and check that the error traps work as expected.

   a. read in a number, multiply it by 3, write out the result

   b. read in a number, square it, write out the result

   c. read in a number, double it then add 1, write out the result

Bibliography

None

Next we look at the primitive data types.