Software Design and Development

4 *The Primitives*

Terry Marris July 2009

Previously, in number input-output, we were introduced to the Integer data type. Now we look at some other data types provided by VB2008.

4.1 The Primitive Data Types

In the last handout, #3, we used the *Integer* data type. *Integer* is part of the VB language and is an example of a primitive data type.

Primitive data types provided by VB2008 include:

type	description	examples
Integer	a whole number	-1, 0, 1
Double	contains a decimal point	3.1416, 99.413, -19.8
Char	just a single character	"A", "a", "!", "2"
String	zero, one or many characters	" ", "X", "tom and jerry"
Boolean	true or false	sun is hot - true

4.2 Integers

You do sums with integers. You can:

- add them: a + b
- multiply them: a * b and
- subtract one from another: a - b

No problem. But dividing one by another ... is a problem.

Look at 7 ÷ 3. The answer is 2.3333. 7 is an integer. 3 is an integer. But 2.3333 is not. So integer division requires a little more thought.

You may remember, when you were very young, doing something like

7 ÷ 3 = 2 remainder 1 (because 2 * 3 = 6, 6 + 1 = 7)

In VB we can do something similar.

```
7 \ 3 = 2
7 Mod 3 = 1
```

So, \ stands for integer division and is called *div*. It gives the integer answer after dividing one integer by another, any decimal part in the answer is cut off, or, as we programmers say, truncated.

Mod gives you just the remainder after dividing one integer by another, anything else is lost.

Exercise 4.1

1. Evaluate (i.e. work out):

- **a.** 11 \ 4
- **b.** 9 \ 3
- **c.** 10 \ 3
- **d.** 24 \ 5
- **e.** 41 \ 8

2. Evaluate

- **a.** 11 Mod 4
- **b.** 9 Mod 3
- **c.** 10 Mod 3
- **d.** 19 Mod 5
- **e.** 27 Mod 8

4.2.1 Div and Mod

Here is a simple problem:

```
read in days
convert days to weeks and days
write out weeks and days
```

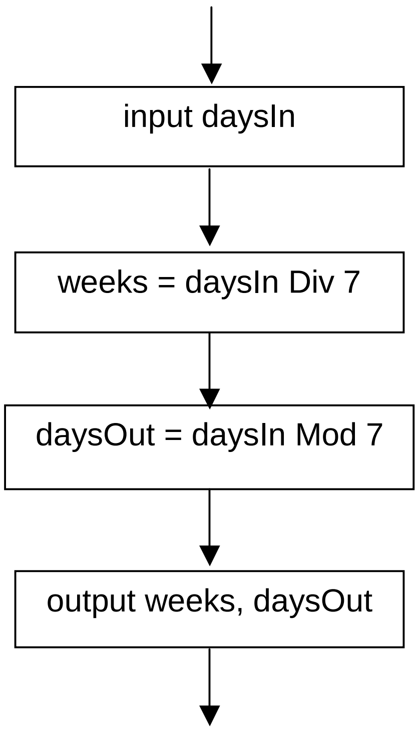
For example, 9 days is 1 week and 2 days.

We add a little more detail to our structured English shown above.

```
read daysIn
weeks = daysIn Div 7
daysOut = daysIn Mod 7
write weeks, daysOut
```

Div? Well, it means \. \ is a VB peculiarity. Div is commonly used to mean integer division in structured English.

For those of us who prefer program flowcharts:



User Interface

1. Project Name = *DivAndMod*

2. File Name = *DivAndMod.vb*

3. Form
   a. Name = *frmDivAndMod*
   b. Text = *Div and Mod*

4. Label
   a. Name = *lblDaysIn*
   b. Text = *Days?*

5. Text Box
   a. Name = *txtDaysIn*

6. Button
   a. Name = *btnOK*
   b. Text = *OK*

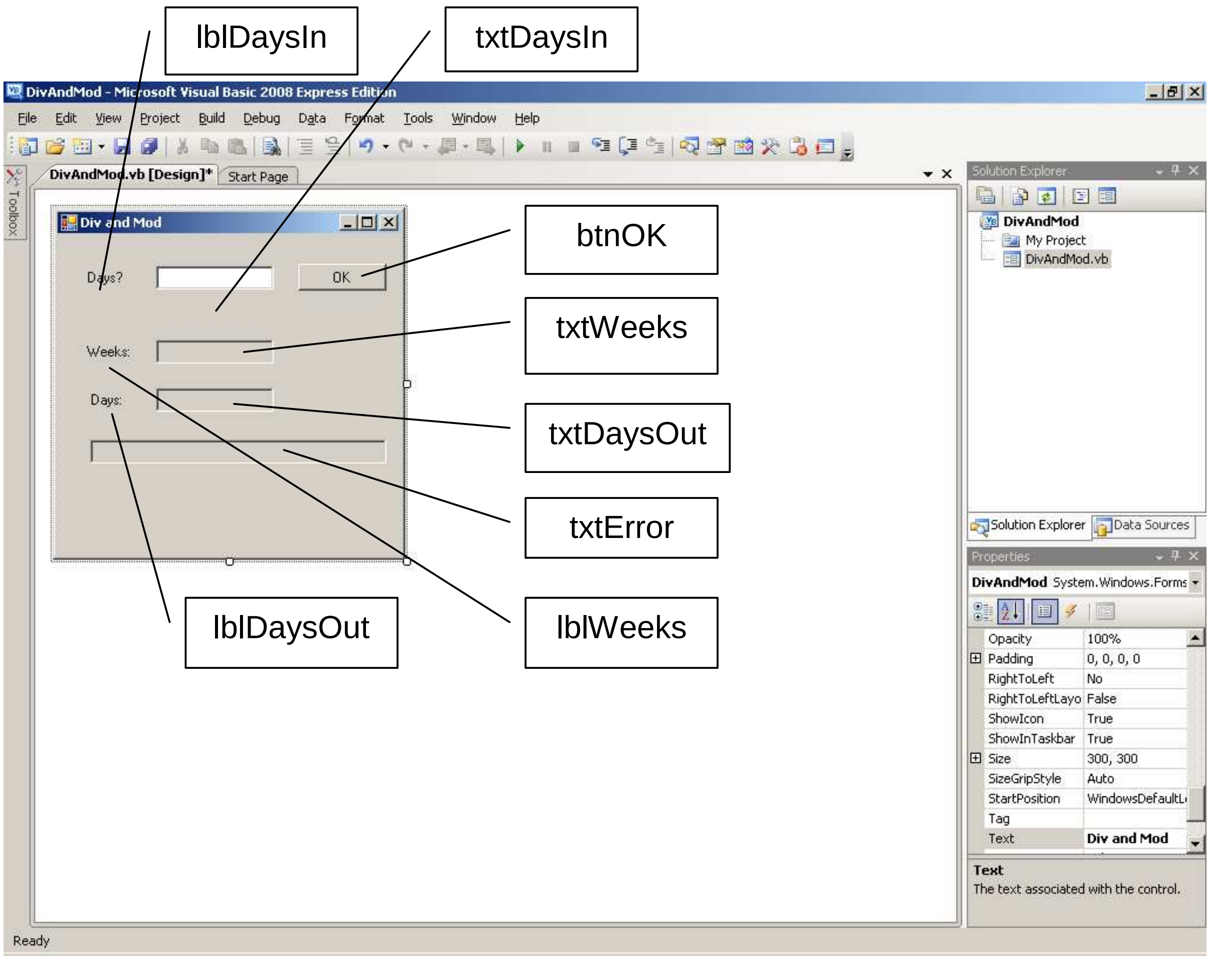
7. Label
   a. Name = *lblWeeks*
   b. Text = *Weeks:*

8. Text Box
   a. Name = *txtWeeks*
   b. Read Only = *True*

9. Label
   a. Name = *lblDaysOut*
   b. Text = *Days:*

10. Text Box
    a. Name = *txtDaysOut*
    b. Read Only = *True*

11. Text Box
    a. Name = *txtError*
    b. Read Only = *True*



Opacity	100%
Padding	0, 0, 0, 0
RightToLeft	No
RightToLeftLayo	False
ShowIcon	True
ShowInTaskbar	True
Size	300, 300
SizeGripStyle	Auto
StartPosition	WindowsDefaultL
Tag	
Text	Div and Mod

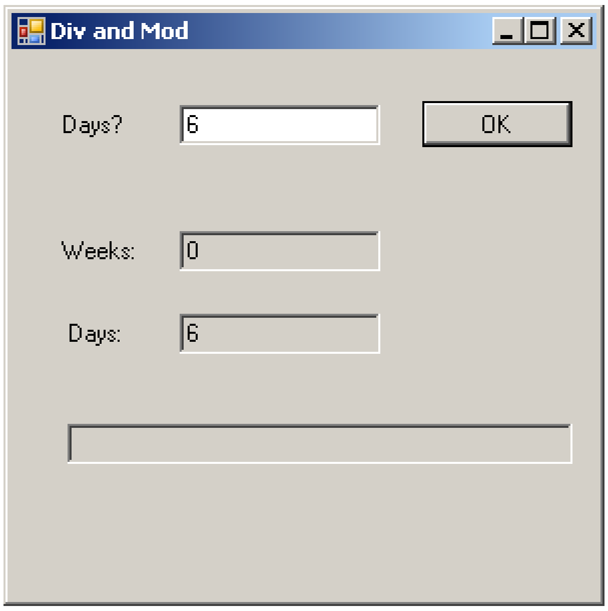
Text
The text associated with the control.

Programmer's Code

12. Under the OK button:

```
Try
    Dim intDaysIn As Integer = Convert.ToInt32(txtDaysIn.Text)
    Dim intWeeks As Integer = intDaysIn \ 7
    Dim intDaysOut As Integer = intDaysIn Mod 7
    txtWeeks.Text = intWeeks.ToString()
    txtDaysOut.Text = intDaysOut.ToString()
Catch ex As FormatException
    txtError.Text = "Error: not whole number"
Catch ex As OverflowException
    txtError.Text = "Error: number too large"
End Try
```

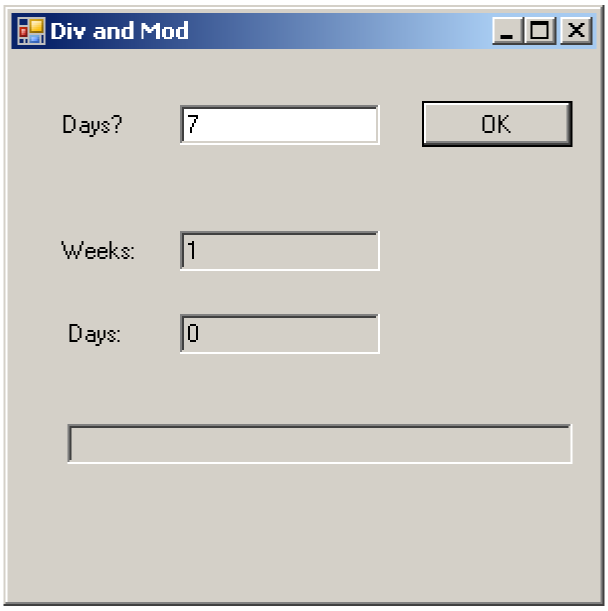
Problem: this program allows you to input a negative number of days, which is a nonsense. This is an error which we acknowledge for now and which we intend to fix later.


Div and Mod

Days? 6 OK

Weeks: 0

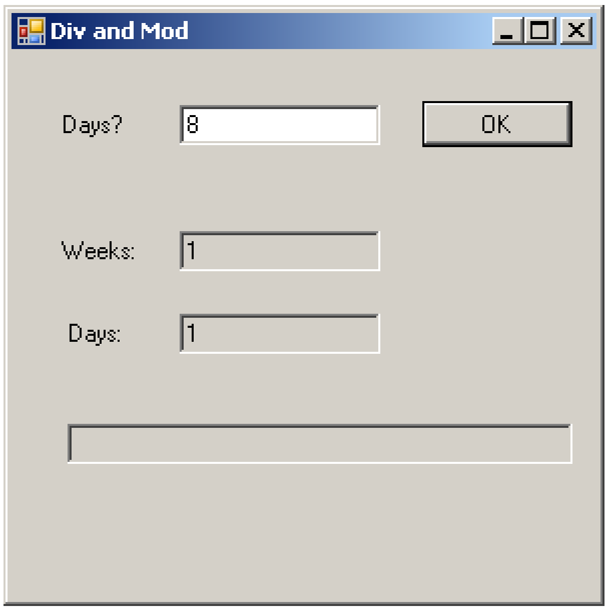
Days: 6


Div and Mod

Days? 7 OK

Weeks: 1

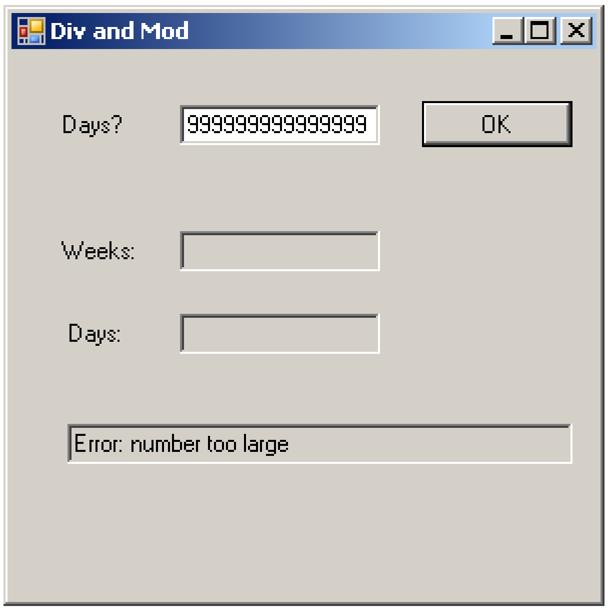
Days: 0


Div and Mod

Days? 8 OK

Weeks: 1

Days: 1

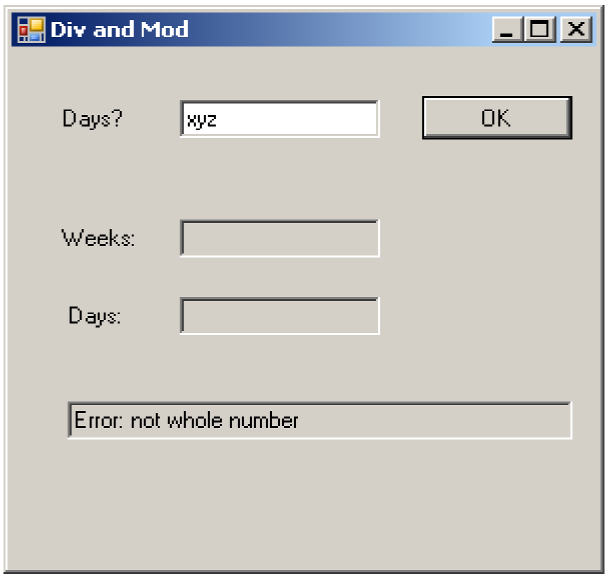

Div and Mod

Days? 999999999999999 OK

Weeks:

Days:

Error: number too large


Div and Mod

Days? xyz OK

Weeks:

Days:

Error: not whole number

Exercise 4.2

1. Try out the program, shown above, that converts a number of days into weeks and days

2. Design, write and test a program that will convert a number of hours into days and hours

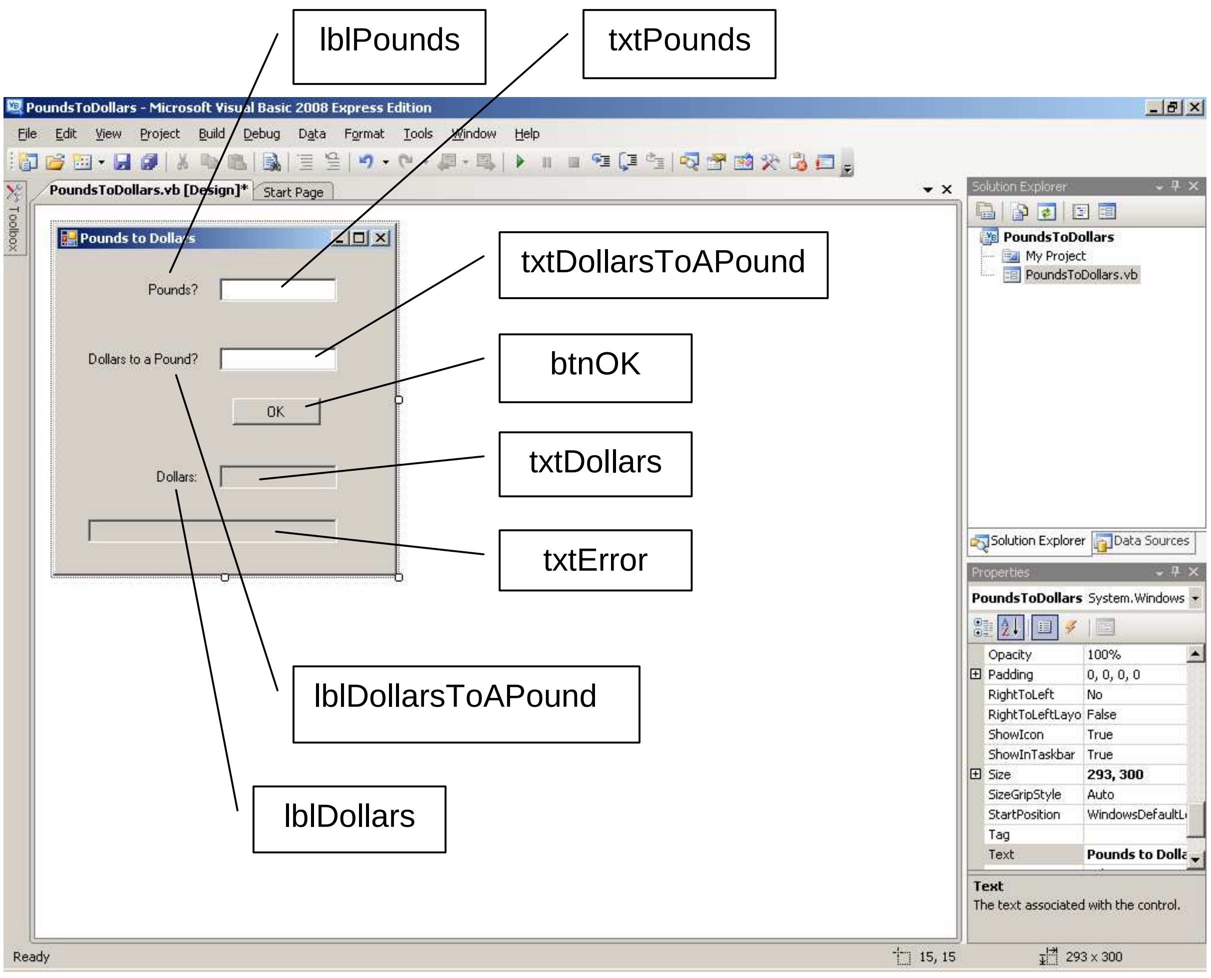
4.3 Doubles

Double stands for double precision floating point. What? It means numbers with a decimal point e.g. 3.1416. You can do the usual arithmetic with values of type Double: +, -, * and / no problem.

Here is a simple problem:

```
read in UK pounds
read in conversionRate to change UK pounds into Australian dollars
convert UK pounds to Australian dollars
write out Australian dollars
```

First, we look at the user interface.



PoundsToDollars System.Windows

Opacity	100%
Padding	0, 0, 0, 0
RightToLeft	No
RightToLeftLayo	False
ShowIcon	True
ShowInTaskbar	True
Size	293, 300
SizeGripStyle	Auto
StartPosition	WindowsDefaultL
Tag	
Text	Pounds to Dolla

Text
The text associated with the control.

User Interface

1. Project Name = *PoundsToDollars*

2. File Name = *PoundsToDollars.vb*

3. Form
    a. Name = *frmPoundsToDollars*
    b. Text = *Pounds to Dollars*

4. Label
    a. Name = *lblPounds*
    b. Text = *Pounds?*

5. TextBox
    a. Name = *txtPounds*

6. Label
    a. Name = *lblDollarsToAPound*
    b. Text = *Dollars to a pound?*

7. TextBox
    a. Name = *txtDollarsToAPound*

8. Button
    a. Name = *btnOK*
    b. Text = *OK*

9. Label
    a. Name = *lblDollars*
    b. Text = *Dollars:*

10. TextBox
    a. Name = *txtDollars*
    b. ReadOnly = *True*

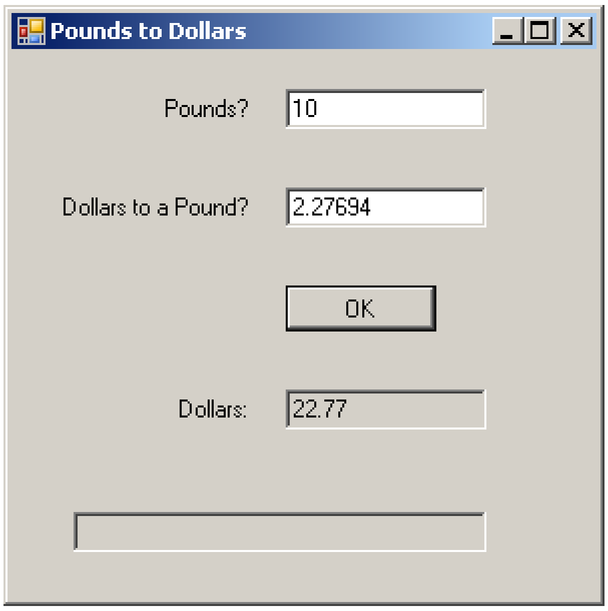
11. TextBox
    a. Name = *txtError*
    b. ReadOnly = *True*

Programming Code

12. Under the OK button

```
Try
    Dim dblPounds As Double = Convert.ToDouble(txtPounds.Text)
    Dim dblDollarsToAPound As Double =
                    Convert.ToDouble(txtDollarsToAPound.Text)
    Dim dblDollars As Double = dblPounds * dblDollarsToAPound
    txtDollars.Text = dblDollars.ToString("F2")
Catch ex As FormatException
    txtError.Text = "Error: not a number"
Catch ex As OverflowException
    txtError.Text = "Error: number too big"
End Try
```

An example of a program run is:



There are 2.27694 dollars to 1 pound.

Therefore 10 pounds = 10 x 2.27694 dollars
= 22.7694 dollars

But the answer shown by the program is 22.77. This is 22.7694 rounded off to two decimal places. How? We format the answer to 2 decimal places by writing

```
dblDollars.ToString("F2")
```

F stands for fixed decimal point - the normal decimal point that you and I use and are familiar with. 2 stands for two digits after the decimal point.

Notice how we used *Convert.ToDouble(...)* to convert our input text into a number of type *Double*.

Exercise 4.3

1. Try out the program, shown above, that converts pounds sterling input to Australian dollars output. Remember to include appropriate comments.

2. Design, write and test a program that will input a number of Australian dollars and output the corresponding number of pounds sterling.

4.4 Chars, Strings and Booleans

We use Chars whenever we want to store a single character such as 'm' or 'f'.

Dim chrGender As Char = "m"

We use Strings whenever we want to store, and perhaps process, a sequence of characters such as text.

Dim strName As String = "Tom and Jerry"

We use Booleans whenever we want to store a *True* or a *False* value.

Dim isFinished = True

I expect we shall have more to say about *Chars*, *Strings* and *Booleans* in later handouts.

Bibliography

http://www.xe.com/ucc accessed 17Sep2008 for currency exchange data

Next we look at selections.