Visual Web Development

Terry Marris September 2008

4 Numbers Input Output

We see how a user may enter numbers and how errors such as invalid number format, overflow and division by zero are trapped.

4.1 Number Types

Fundamental Visual Basic number types include:

- Integer: whole numbers such as ..., -2, -1, 0, 1, 2, 3, ...
- Double: decimal fraction numbers such as: 3.1416, 57.296, 12.00, -0.01

We can add (+) subtract (-) and multiply (*) no problem. Division of one integer by another often gives a none-integer result. For example:

1 / 2 = 0.5

But, if we require an integer result after dividing one integer by another, for example:

7 ÷ 3 = 2 remainder 1   (because 2 x 3 + 1 = 7)

we use the div and mod operators, \ and Mod respectively. So:

7 \ 3	=	2	remainder is cut off
7 Mod 3	=	1	remainder after integer division

We remember that division by zero is not defined. So both

7 \ 0 and 7 Mod 0

have no defined answer; they are indeterminate.

Double stands for double precision floating point and represents real numbers (i.e. those with a decimal point).

4.2 Exercise 1

1 Evaluate:

a 16 \ 3 **b** 20 \ 4 **c** 5 \ 5 **d** 4 \ 5 **e** 7 \ 5

2 Evaluate:

a 16 Mod 3 **b** 20 Mod 4 **c** 5 Mod 5 **d** 4 Mod 5 **e** 7 Mod 5

4.3 Precedence

Precedence is the order in which operations are carried out. In Arithmetic:

brackets first; they have the highest priority

then *, /, \ and Mod

then + and - last, both have the lowest priority

For example:

(40 - 32) * 5 \ 9	=	8 * 5 \ 9	[brackets first]
	=	40 \ 9	[multiply]
	=	4	integer division]

4.4 Exercise 2

1 Evaluate

a 5 + 7 * 9 **b** (5 + 7) * 9 **c** 5 \ 9 * (212 - 32)

d 5.0 / 9.0 * (212.0 - 32.0)

4.5 Gear Ratios

Many of us have ridden bicycles with gears. The gear ratio of a particular combination of wheel size, front chain wheel and rear sprocket is given by:

gearRatio = wheelDiameter x chainWheel / rearSprocket

For example, if

wheelDiameter	=	27 inches
chainWheel	=	52 teeth
rearSprocket	=	14 teeth

then

gearRatio	=	27 x 52 / 14 inches
	=	100.3 inches

The lower the ratio, the easier it is to pedal, especially useful for getting you up hills.

Typical wheel diameters are 27, 26 and 24 inches. Typical chain wheel sizes are between 52 and 22 teeth. Typical sprocket sizes are between 13 and 34 teeth.

4.6 Use Case

Use Case	GearRatioCalculator
Goal	to display the gear ratio for a combination of wheel, chain wheel and sprocket
Pre-condition	**1** integer values for wheel size (in inches), number of teeth on chain wheel, number of teeth on sprocket, are entered **2** sprocket entered is not zero
Post-condition	the gear ratio in inches is displayed
Initiating Actor	the user

Main Success Scenario

1 system prompts user for wheel, chain wheel and sprocket sizes
2 user enters wheel, chain wheel and sprocket sizes
3 system calculates and displays gear ratio
4 exit success

Exceptions

2a non-numeric input
  1 system displays error message
  2 resume 2
2b zero rear sprocket size
  1 system displays error message
  2 resume 2
2c overflow
  1 system displays error message
  2 resume 2

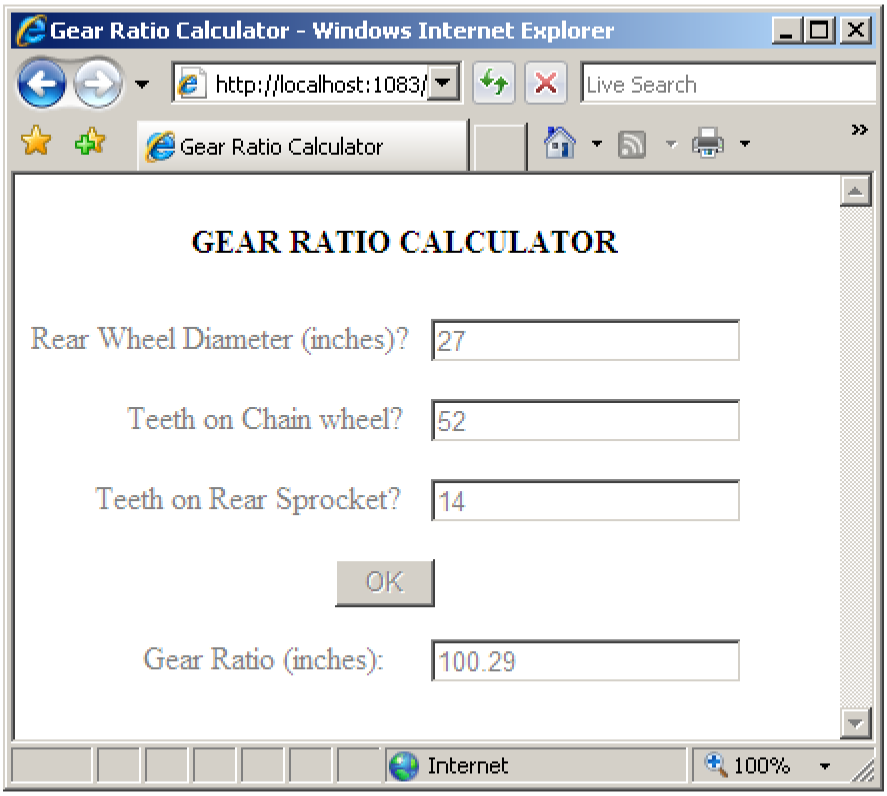
Non-numeric input occurs when, for example, the user does not enter digits where digits are expected. For example, they might enter the letter capital I for the digit 1, or a space between 2 and 7 when they intended 27.

Since division by zero is not defined it is up to us programmers to warn users if they try to divide by zero.

A computer's memory is finite, and so is the size of numbers it can store. Overflow occurs if either the number entered, or the result of a calculation, is too large to be stored.

4.7 User Interface

The user interface is shown below.



GEAR RATIO CALCULATOR

Rear Wheel Diameter (inches)? 27

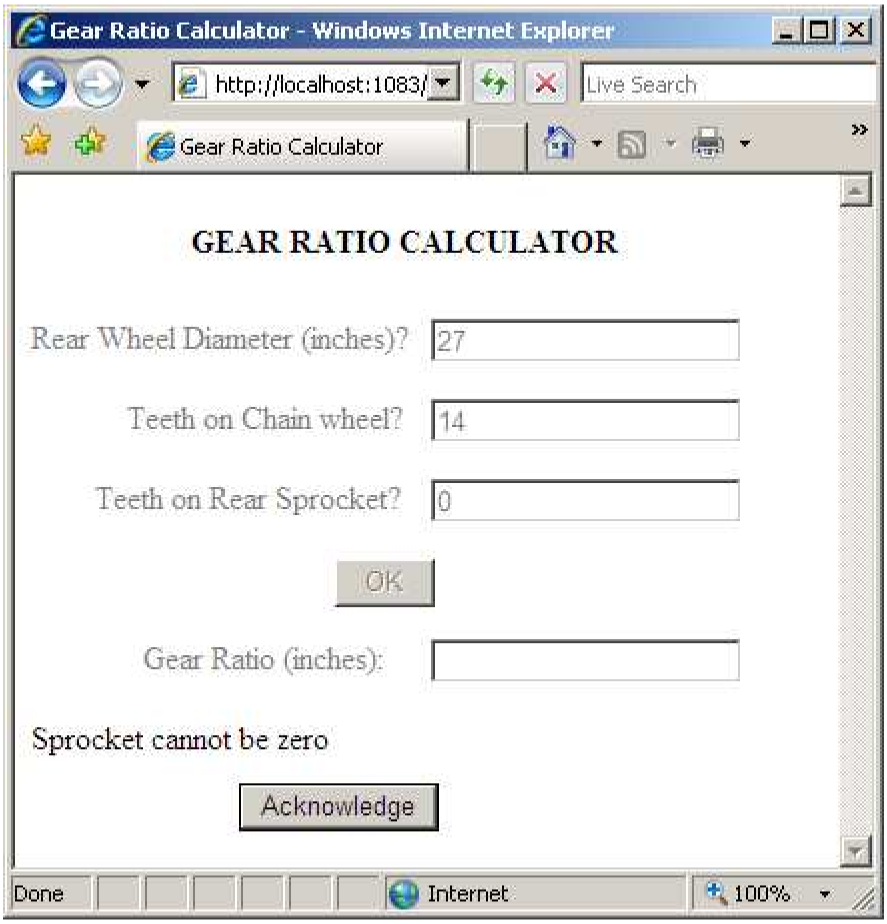
Teeth on Chain wheel? 52

Teeth on Rear Sprocket? 14

Gear Ratio (inches): 100.29

Fig 4.1 The normal, straightforward, successful case

User enters the rear wheel diameter, the number of teeth on the chainwheel and on the rear sprocket, and presses OK. The computer responds by displaying the gear ratio.



GEAR RATIO CALCULATOR

Rear Wheel Diameter (inches)? 27

Teeth on Chain wheel? 14

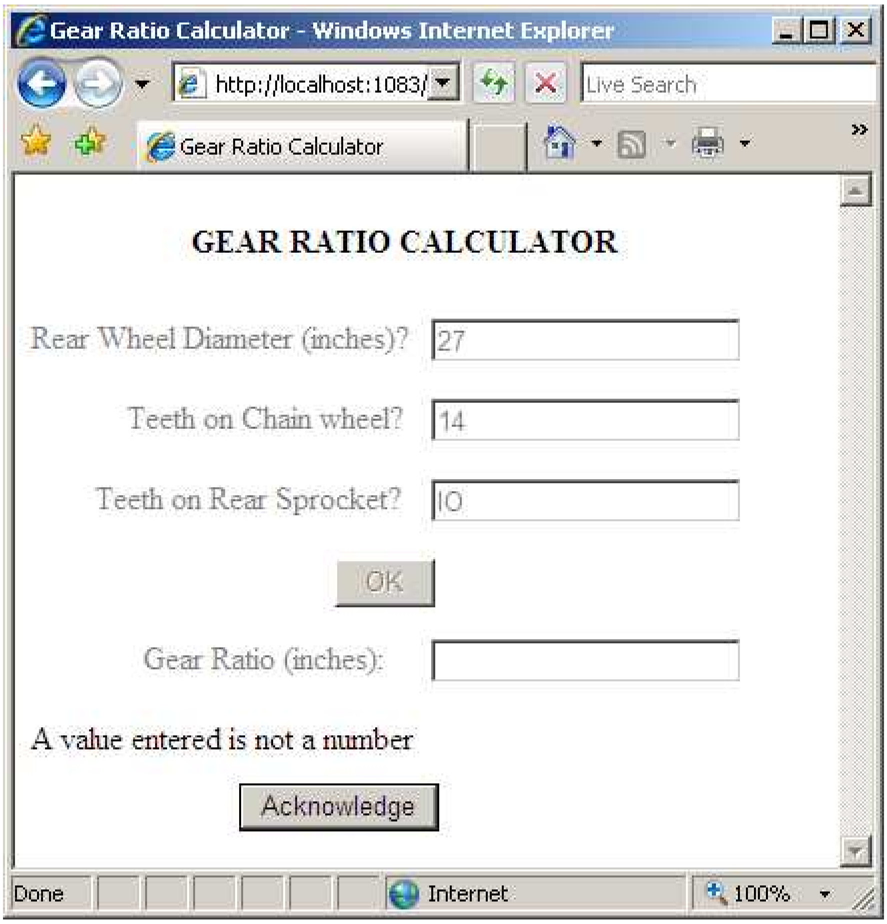
Teeth on Rear Sprocket? 0

Gear Ratio (inches):

Sprocket cannot be zero

Fig 4.2 Cannot divide by zero

User enters zero for teeth on rear sprocket; the computer responds with the error message: Sprocket cannot be zero.



GEAR RATIO CALCULATOR

Rear Wheel Diameter (inches)? 27

Teeth on Chain wheel? 14

Teeth on Rear Sprocket? IO

OK

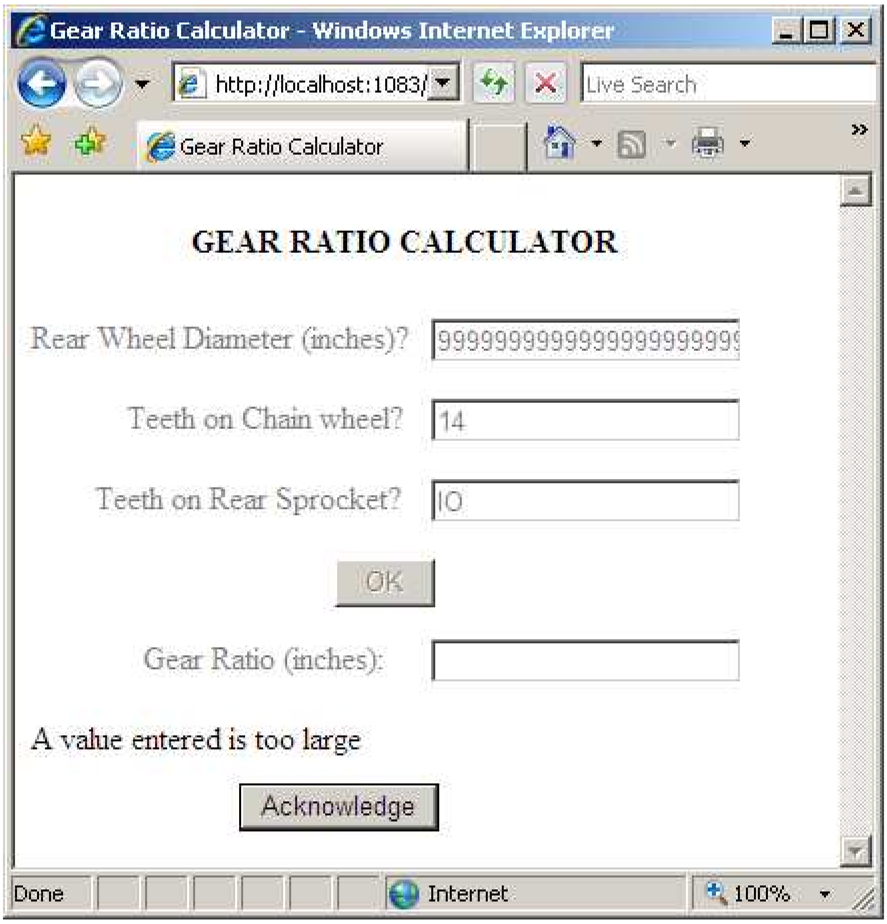
Gear Ratio (inches):

A value entered is not a number

Acknowledge

Fig 4.3 IO is not a number (but 10 is)

User enters the letters IO, computer responds with the error message: A value entered is not a number.



GEAR RATIO CALCULATOR

Rear Wheel Diameter (inches)? 9999999999999999999999

Teeth on Chain wheel? 14

Teeth on Rear Sprocket? IO

Gear Ratio (inches):

A value entered is too large

Fig 4.4 9999999999999999999999999 is too large a number

User enters a ridiculously large number, the computer responds with an error message.

4.8 Properties and Values

We create the user interface and provide property values as shown below.

Control	Property	Value
Form	File Name	NumbersIO.aspx
Document	Title	Gear Ratio Calculator
Label	ID Text Font Bold	lblTitle Gear Ratio Calculator True
Label	ID Text	lblWheelDiameter Rear Wheel Diameter (inches)?
Label	ID Text	lblChainwheel Teeth on Chain wheel?
Label	ID Text	lblSprocket Teeth on Rear Sprocket?
Label	ID Text	lblGear Gear Ratio (inches)
Label	ID Text	lblError <blank>
Text Box	ID Text	txtWheelDiameter <blank>
Text Box	ID Text	txtChainwheel <blank>
Text Box	ID Text	txtSprocket <blank>
Text Box	ID Text	txtGear <blank>
Button	ID Text Width	btnOK OK 50px
Button	ID Text Width	btnAck Acknowledge 100px

4.9 Parameters and Arguments

The procedures to enable and disable input and output, and to show the hide and show the error interface, are straightforward.

```
    Sub EnableInput()
        lblWheelDiameter.Enabled = True
        txtWheelDiameter.Enabled = True
        lblChainwheel.Enabled = True
        txtChainwheel.Enabled = True
        lblSprocket.Enabled = True
        txtSprocket.Enabled = True
        btnOK.Enabled = True
    End Sub

    Sub DisableInput()
        lblWheelDiameter.Enabled = False
        txtWheelDiameter.Enabled = False
        lblChainwheel.Enabled = False
        txtChainwheel.Enabled = False
        lblSprocket.Enabled = False
        txtSprocket.Enabled = False
        btnOK.Enabled = False
    End Sub

    Sub EnableOutput()
        lblGear.Enabled = True
        txtGear.Enabled = True
    End Sub

    Sub DisableOutput()
        lblGear.Enabled = False
        txtGear.Enabled = False
    End Sub

    Sub ShowError()
        lblError.Visible = True
        btnAck.Visible = True
    End Sub

    Sub HideError()
        lblError.Visible = False
        btnAck.Visible = False
    End Sub
```

The ProcessError procedure receives an error message to be displayed; this message is identified by ByVal errorMessage As String.

ByVal errorMessage As String is known as a parameter.

```
Sub ProcessError(ByVal errorMessage As String)
    DisableInput()
    DisableOutput()
    ShowError()
    lblError.Text = errorMessage
    btnAck.Focus()
End Sub
```

To use the ProcessError procedure, we provide its name along with an error message such as "Sprocket cannot be zero".

```
  If intSprocket = 0 Then
      ProcessError("Sprocket cannot be zero")
  End If
```

The error message, "Sprocket cannot be zero" is passed to errorMessage.

"Sprocket cannot be zero" is known as an argument. Arguments are passed to parameters for processing.

The argument-parameter pair is the mechanism by which data is passed into procedures.

4.10 Try .. Catch ...

Try ... Catch is the Visual Basic mechanism for trapping (some) run-time errors. For example:

```
Try
  intWheelDiameter = Convert.ToInt32(txtWheelDiameter.Text)
  ...
  ...
  ...
Catch ex As FormatException
  ProcessError("A value entered is not a number")
End Try
```

Here, we try to convert the text input into an integer. If the (idiot) user enters non-numeric data, the error is trapped as a FormatException and we then say: go and process the error.

4.11 Converting Text to Numbers

There are various methods of converting text input to integer; one is the Convert.ToInt32() method. This method allows us to use the Try ... Catch mechanism.

The Convert.toDouble() method converts text input to values of type double.

4.12 Declaring Variables

A variable is a location in memory. It has a name, a type and a value.

```
Dim intWheelDiameter As Integer = 0
```

A declaration, as shown above, introduces the properties of a variable and VB reserves a place in memory for it.

4.13 Formatting Double Output

If the output is something like 23.65841 we might like to format it to 2 decimal places. This is what

```
txtGear.Text = String.Format("{0:F}", dblGear)
```

does. "{0:F}" is an example of a *format string*.

4.14 The Code

The entire code is shown below.

```
Partial Class _Default
    Inherits System.Web.UI.Page

    Sub EnableInput()
        lblWheelDiameter.Enabled = True
        txtWheelDiameter.Enabled = True
        lblChainwheel.Enabled = True
        txtChainwheel.Enabled = True
        lblSprocket.Enabled = True
        txtSprocket.Enabled = True
        btnOK.Enabled = True
    End Sub

    Sub DisableInput()
        lblWheelDiameter.Enabled = False
        txtWheelDiameter.Enabled = False
        lblChainwheel.Enabled = False
        txtChainwheel.Enabled = False
        lblSprocket.Enabled = False
        txtSprocket.Enabled = False
        btnOK.Enabled = False
    End Sub

    Sub EnableOutput()
        lblGear.Enabled = True
        txtGear.Enabled = True
    End Sub

    Sub DisableOutput()
        lblGear.Enabled = False
        txtGear.Enabled = False
    End Sub
```

The procedures to enable and disable the input and output are straightforward

```
    Sub ShowError()
        lblError.Visible = True
        btnAck.Visible = True
    End Sub

    Sub HideError()
        lblError.Visible = False
        btnAck.Visible = False
    End Sub
```

The procedures to hide and show the error interface are also straighforward.

```
    Sub ProcessError(ByVal errorMessage As String)
        DisableInput()
        DisableOutput()
        ShowError()
        lblError.Text = errorMessage
        btnAck.Focus()
    End Sub
```

ProcessError receives a String value as a parameter and assigns it to lblError.Text as well as calling upon other procedures to perform their tasks.

```
    Protected Sub Page_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
        EnableInput()
        DisableOutput()
        HideError()
        txtWheelDiameter.Focus()
    End Sub
```

On start up we enable the input, disable the output and hide the error mechanism on the screen

Most of the action occurs when the OK button is clicked

```
    Protected Sub btnOK_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btnOK.Click
        Dim intWheelDiameter As Integer = 0
        Dim intChainwheel As Integer = 0
        Dim intSprocket As Integer = 0
        Dim dblGear As Double = 0.0
```

We declare variables, give them their initial values, and convert the text input to integer values

```
        Try
            intWheelDiameter = Convert.ToInt32(txtWheelDiameter.Text)
            intChainwheel = Convert.ToInt32(txtChainwheel.Text)
            intSprocket = Convert.ToInt32(txtSprocket.Text)

            If intSprocket = 0 Then
                Throw New ApplicationException()
            End If
```

An error case

```
            dblGear = intWheelDiameter * intChainwheel / intSprocket
            txtGear.Text = String.Format("{0:F}", dblGear)
            DisableInput()
            DisableOutput()
```

The non-error case

```
        Catch ex As ApplicationException
            ProcessError("Sprocket cannot be zero")
        Catch ex As OverflowException
            ProcessError("A value entered is too large")
        Catch ex As FormatException
            ProcessError("A value entered is not a number")
        End Try
    End Sub
```

Dealing with the error cases

```
    Protected Sub btnAck_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btnAck.Click
        HideError()
        EnableInput()
        txtWheelDiameter.Focus()
    End Sub
End Class
```

When the Acknowledge button is clicked we set the screen up for input

4.15 Exercise 3

1. Try out the program, shown above, that prompts the user to enter the rear wheel size, and the number of teeth on a chainwheel and rear sprocket, and outputs the gear ratio in inches.

2. Enhance the error case management for the gear calculator program shown above. It is an error if:

   - **a.** the wheel size is less than 24 or more than 27
   - **b.** the chainwheel size is less than16 or more than 56
   - **c.** the sprocket size is less than 12 or more than 34

3. Write a program that converts degrees Celsius into degrees Fahrenheit. You might like to use the fact that

   degreesCelsius = (degreesFarenheit - 32) * 5 / 9

   and 32ºF = 0ºC and 212ºF = 100ºC

4. Write a program that displays the real roots of a quadratic equation. You might like to use the fact that if

$$ax^2 + bx + c = 0 \text{ then } x = \frac{-b \pm \sqrt{(b^2 - 4ac)}}{2a}$$