My first programme

```
Imports System


Module Program
    Sub Main(args As String())
        ' This is my first program
        Console.WriteLine("How are you?")
    End Sub
End Module
```

Numeric data types

```
Module Program
    Sub Main(args As String())
        ' This is my first program
        Dim val1 As Integer = 2
        Dim val2 As Integer = 3
        Console.WriteLine("My numbers are " & val1 & " and " & val2 & "." _
            & vbCrLf & "The total is " & (val1 + val2))
    End Sub
End Module
' Byte 0 to 255                          ( 3 digits) - 1 byte
' Short -32,768 to +32,767               ( 5 digits) - 2 bytes
' UShort      0 to +65,535
' Integer -2 billion to +2 billion       (10 digits) - 4 bytes
' Long    -9 quintillion + 9 quintillion (19 digits) - 8 bytes
' BaSIL   LISB
```

Other data types

```
    Sub Main(args As String())
        ' This is my first program
        'Dim val1 As Decimal = 123456789123456789.1234D
        'Dim val2 As Decimal = 0
        'Console.WriteLine("My numbers are " & val1 & " and " & val2 & "." _
        '    & vbCrLf & "The total is " & (val1 + val2))
        Dim val3 As String = "This is my string"
        val3 = val3 & " and this is also my string"
        'val3 &= " and this is also my string" 'Alternative way
        Console.WriteLine(val3)
        Dim val4 As Char = "Y"c
```

```
        Console.WriteLine(val4)
        Dim val5 As Date = #2030/12/1 13:45:12#
        Console.WriteLine(val5)
        Dim val6 As Boolean = True
        Console.WriteLine(val6)
    End Sub
```

If Statement

```
    Sub Main(args As String())
        ' This is my first program
        Dim val1 As Integer = 2
        If val1 = 2 Then
            Console.WriteLine("This occurs is the condition is true.")
            Console.WriteLine("This condition has been met.")
        End If
        Console.WriteLine("This is the end of the program.")
    End Sub
```

If statement and the And, Or and Not operators

```
    Sub Main(args As String())
        ' This is my first program
        Dim val1 As Integer = 2
        Dim val2 As Integer = 3
        Dim val3 As Integer = 4
        ' mathematical symbols, conditions, operators
        If Not (val3 < val2 And val2 > val1) Then
            Console.WriteLine("This occurs is the condition is true.")
            Console.WriteLine("This condition has been met.")
        End If
        Console.WriteLine("This is the end of the program.")
    End Sub
```

Dividing integers

```
        Dim val1 As Double = 14.2
        Dim val2 As Double = -4.8
        Dim val3 As Integer = 4
        ' mathematical symbols, conditions, operators
        Console.WriteLine(Fix(val2)) ' 4.8 truncates -> 4
        Console.WriteLine(Int(val2)) ' 4.8 rounded down -> 4
```

```
Console.WriteLine(CInt(val2))  ' converts 4.8 -> integer. rounding 4.8 -> 5
```

if…else statement

```
Dim val1 As Double = 2
Dim val2 As Double = 1
If val2 > val1 Then
    Console.WriteLine("This condition has been met.")
Else
    If val2 = val1 Then
        Console.WriteLine("The values are the same.")
    Else
        Console.WriteLine("This condition has not been met.")
    End If
End If
Console.WriteLine("This is the end of the program.")
```

Select [Case]…Case statement

```
Dim val1 As Double = 17
Dim val2 As Double = 10
Select Case val1 Mod val2
    Case 1
        Console.WriteLine("This answer is one")
    Case 1, 2
        Console.WriteLine("The answer is two")
    Case 3 To 5
        Console.WriteLine("The answer is three to five")
    Case 6, 8
        Console.WriteLine("The answer is six or eight")
    Case Else
        Console.WriteLine("The answer is something else.")
End Select
Console.WriteLine("This is the end of the program.")
```

For loop

```
Dim val2 As Double = 4
For val1 As Decimal = 1 To 20 Step 2
    If val1 = 10 Then
        Exit For
    End If
    Console.WriteLine(val1 & " modulus " & val2 & " is " & (val1 Mod val2) & ".")
Next
Console.WriteLine("This is the end of the program.")
```

while and do…while loop

```
Dim val1 As Integer = 30
Dim val2 As Double = 4
While val1 <= 20
    If val1 = 100 Then
        Exit While
    End If
    Console.WriteLine(val1 & " modulus " & val2 & " is " & (val1 Mod val2) & ".")
    val1 = val1 + 1
End While
Console.WriteLine("This is the end of the program.")


val1 = 30
Do
    If val1 = 100 Then
        Exit Do
    End If
    Console.WriteLine(val1 & " modulus " & val2 & " is " & (val1 Mod val2) & ".")
    val1 = val1 + 1
Loop While val1 <= 20
Console.WriteLine("This is the end of the program.")
```

var1+=

```
Dim val1 As Integer = 30
Dim val2 As Double = 4
Do
    If val1 = 100 Then
        Exit Do
    End If
    Console.WriteLine(val1 & " modulus " & val2 & " is " & (val1 Mod val2) & ".")
    val1 += 10
Loop While val1 >= 20
Console.WriteLine("This is the end of the program.")
```

Structure exception handling

```
Dim val1 As Integer = 30
Dim val2 As Double
Dim UserInput As String
Console.Write("Please enter a denominator: ")
```

```
        UserInput = Console.ReadLine()
        Try
            val2 = CDbl(UserInput)
            Console.WriteLine(val1 & " modulus " & val2 & " is " & (val1 Mod val2) & ".")
        Catch ex As FormatException
            Console.WriteLine("There is a format exception.")
        Catch ex As Exception
            Console.WriteLine("There is a problem with your input.")
            Throw
        Finally
            Console.WriteLine("The end.")
        End Try            'StackOverflowException
```

Classes and Sub procedures

```
Module Program
    Sub Main(args As String())
        Dim Item = New Computer
        Item.Typing()
        Item.Sound()
    End Sub
End Module

Class Computer
    Sub Typing()
        Console.WriteLine("I am typing")
    End Sub
    Sub Sound()
        Console.WriteLine("I am creating a sound")
    End Sub
End Class
```

Auto-implemented Properties

```
Module Program
    Sub Main(args As String())
        Dim SoundOfTyping As String
        Dim Item = New Computer
        Item.TypeOfComputer = "Laptop"
        SoundOfTyping = Item.Typing()
        Console.WriteLine(SoundOfTyping & " goes the " & Item.TypeOfComputer & ".")
        Item.Sound()
```

```
        Dim Item2 = New Computer
        Item2.TypeOfComputer = "Desktop"
        SoundOfTyping = Item2.Typing()
        Console.WriteLine(SoundOfTyping & " goes the " & Item2.TypeOfComputer & ".")
        Item2.Sound()
        Console.WriteLine(SoundOfTyping & " goes the " & Item.TypeOfComputer & ".")
    End Sub
End Module

Class Computer
    Property TypeOfComputer As String
    Function Typing() As String
        Console.WriteLine("I am typing")
        Return "type type type"
    End Function
    Sub Sound()
        Console.WriteLine("I am creating a sound")
    End Sub
End Class
```

Implementing properties

```
Class Computer
    Private _TypeOfComputer As String
    Public Property TypeOfComputer() As String
        Get
            Select Case _TypeOfComputer
                Case "Laptop"
                    Return "pretty good laptop"
                Case "Desktop"
                    Return "nice desktop"
                Case Else
                    Return _TypeOfComputer
            End Select
        End Get
        Set(ByVal value As String)
            Select Case value
                Case "Laptop", "Desktop"
                    _TypeOfComputer = value
                Case Else
                    _TypeOfComputer = "Unknown"
            End Select
        End Set
```

```
    End Property
    Function Typing() As String
        Console.WriteLine("I am typing")
        Return "type type type"
    End Function
    Sub Sound()
        Console.WriteLine("I am creating a sound")
    End Sub
End Class
```

Using New and setting parameters

```
        Dim Item = New Computer() With {.TypeOfComputer = "Laptop"}
```

Creating a constructor

```
        Dim Item = New Computer("Laptop")

Class Computer
    Public Sub New()
        Console.WriteLine("I am creating a new computer.")
    End Sub
    Public Sub New(strTypeOfComputer As String)
        Me.New()
        TypeOfComputer = strTypeOfComputer
    End Sub
```

Events

```
Module Program
    Sub Main(args As String())
        Dim SoundOfTyping As String
        Dim Item = New Computer("Laptop")
        AddHandler Item.IamTyping, AddressOf IamTyping_EventHandler
        With Item
            SoundOfTyping = .Typing()
            Console.WriteLine(SoundOfTyping & " goes the " & .TypeOfComputer & ".")
            .Sound()
        End With
    End Sub
    Sub IamTyping_EventHandler()
        Console.WriteLine("I am typing in an event.")
    End Sub
End Module
```

```
Class Computer
    Public Event IamTyping()
    Function Typing() As String
        'Console.WriteLine("I am typing")
        RaiseEvent IamTyping()
        Return "type type type"
    End Function
End Class
```

Understanding core programming

Recursion

```
        Console.WriteLine(GrandTotal(9))
    End Sub

    Dim RunningTotal As Integer
    Function GrandTotal(StartWith As Integer)
        RunningTotal = 0
        Return AddUp(StartWith)
    End Function

    Function AddUp(AddNumber As Integer)
        RunningTotal += AddNumber
        If AddNumber <= 0 Then
            Return RunningTotal
        End If
        Return AddUp(AddNumber - 1)

    End Function
```

```
        Console.WriteLine(GrandTotal(9))
    End Sub

    Function GrandTotal(AddNumber As Integer, Optional RunningTotal As Integer = 0)
        RunningTotal += AddNumber
        If AddNumber <= 0 Then
            Return RunningTotal
        End If
        Return GrandTotal(AddNumber - 1, RunningTotal)
    End Function
```

Understanding object-oriented programming

Starting inheritance

```
Module Program
    Sub Main(args As String())
        Console.WriteLine("COMPUTER")
        Dim Item As New Computer()
        Item.TypeOfComputer = "A generic computer"
        Console.WriteLine(Item.Hardware())
        Console.WriteLine(Item.TypeOfComputer)
        Console.WriteLine()

        Console.WriteLine("LAPTOP")
        Dim Item2 As New Laptop()
        Item2.TypeOfComputer = "A laptop"
        Console.WriteLine(Item2.Hardware())
        Console.WriteLine(Item2.TypeOfComputer)
        Console.WriteLine()
    End Sub
End Module


Class Computer 'Base, parent class - Laptop IS A Computer
    Public Property TypeOfComputer As String
    Function Hardware()
        Return "Hardware"
    End Function
End Class

Class Laptop 'Derived, child class. Inheritence
    Inherits Computer
End Class
```

More inheritance

```
Module Program
    Sub Main(args As String())
        Console.WriteLine("COMPUTER")
        Dim Item As New Computer()
        Item.TypeOfComputer = "A generic computer"
        Console.WriteLine(Item.Hardware())
        Console.WriteLine(Item.TypeOfComputer)
```

```
        Console.WriteLine()

        Console.WriteLine("LAPTOP")
        Dim Item2 As New Laptop()
        Item2.TypeOfComputer = "A laptop"
        Console.WriteLine(Item2.Hardware())
        Console.WriteLine(Item2.TypeOfComputer)
        Console.WriteLine(Item2.AmIMobile)
        Console.WriteLine()

        Console.WriteLine("Desktop")
        Dim Item3 As New Desktop()
        Item3.TypeOfComputer = "A desktop"
        Console.WriteLine(Item3.Hardware())
        Console.WriteLine(Item3.TypeOfComputer)
        Console.WriteLine(Item3.AmIMobile)
        Console.WriteLine()

    End Sub
End Module


Class Computer 'Base, parent class - Laptop IS A Computer
    Public Property TypeOfComputer As String
    Function Hardware()
        Return "Hardware"
    End Function
End Class

Class Laptop 'Derived, child class. Inheritence
    Inherits Computer
    Function AmIMobile()
        Return "Yes, I am mobile"
    End Function
End Class

Class Desktop
    Inherits Computer
    Function AmIMobile()
        Return "No, I am not mobile"
    End Function
End Class
```

Polymorphism

```
MustInherit Class Computer 'Base, parent class - Laptop IS A Computer
    Public Property TypeOfComputer As String
    Function Hardware()
        Return "Hardware"
    End Function
    MustOverride Function AmIMobile()
End Class

Class Laptop 'Derived, child class. Inheritence
    Inherits Computer
    Overrides Function AmIMobile()
        Return "Yes, I am mobile"
    End Function
End Class
```

```
Class Computer 'Base, parent class - Laptop IS A Computer
    Public Property TypeOfComputer As String
    Function Hardware()
        Return "Hardware"
    End Function
    Overridable Function AmIMobile()
        Return "I don't know"
    End Function
End Class

Class Laptop 'Derived, child class. Inheritence
    Inherits Computer
    Overrides Function AmIMobile()
        Return "Yes, I am mobile"
    End Function
End Class
```

```
Class Computer 'Base, parent class - Laptop IS A Computer
    Public Property TypeOfComputer As String
    Function Hardware()
        Return "Hardware"
    End Function
    Function AmIMobile()
        Return "I don't know"
    End Function
```

```
End Class

Class Laptop 'Derived, child class. Inheritence
    Inherits Computer
    Shadows Function AmIMobile()
        Return "Yes, I am mobile"
    End Function
End Class
```

What is the difference between "Shadows" and "Overrides"

```
Class Computer
    Public Property TypeOfComputer As String
    Function Hardware()
        Return "Hardware"
    End Function
    Overridable Function AmIMobile()
        Return "I don't know"
    End Function
End Class

Class Laptop 'Derived, child class. Inheritence
    Inherits Computer
    Overrides Function AmIMobile()
        Return "Yes, I am mobile"
    End Function
End Class

        Dim Item4 As Computer = Item2
        Console.WriteLine(Item4.AmIMobile()) Result: Yes, I am mobile – from the Laptop class
```

```
Class Laptop 'Derived, child class. Inheritence
    Inherits Computer
    Shadows Function AmIMobile()
        Return "Yes, I am mobile"
    End Function
End Class

        Dim Item4 As Computer = Item2
        Console.WriteLine(Item4.AmIMobile()) Result: Yes, I am mobile – from the Computer class
```

NotOverridable Classes

```
Class Laptop 'Derived, child class. Inheritence
    Inherits Computer
    NotOverridable Overrides Function AmIMobile()
        Return "Yes, I am mobile"
    End Function
End Class

Class LaptopSmall
    Inherits Laptop
    Overrides Function AmIMobile() 'Does not work.
        Return "Yes, I am very mobile"
    End Function
End Class
```

Encapsulation

```
    Private strTypeOfComputer As String
    Public Property TypeOfComputer As String
        Get
            Return strTypeOfComputer
        End Get
        Set(value As String)
            strTypeOfComputer = value
        End Set
    End Property
```

Understanding general software development

Single-dimension Arrays

```
    Sub Main(args As String())
        'Dim items As Integer() = {10, 11, 12, 13, 14, 15, 16, 17}
        Dim items(4) As Integer
        items(0) = 10
        items(1) = 13
        items(3) = 13
        items.SetValue(14, 4)
        Console.WriteLine("The first item is " & items(0))
        Console.WriteLine("The last item is " & items(4))
        Console.WriteLine("The number of items is {0}", items.Length)
        For x As Integer = 0 To items.Length - 1
```

```
            Console.WriteLine(items(x))
            If items(x) = 13 Then
                Console.WriteLine("The condition is in item {0}", x)
            End If
        Next
        For Each itm As Integer In items
            Console.WriteLine(itm)
        Next
    End Sub
```

Multi-dimension arrays

```
Module Program
    Sub Main(args As String())
        'Dim items As Integer() = {10, 11, 12, 13, 14, 15, 16, 17}
        Dim items(0 To 4, 0 To 1) As Integer
        items(0, 0) = 10
        items(1, 0) = 13
        items(3, 1) = 13
        items.SetValue(14, 4, 0)
        Console.WriteLine("The first item is " & items(0, 0))
        Console.WriteLine("The last item is " & items(4, 1))
        Console.WriteLine("The number of items is {0}", items.Length)
        Console.WriteLine("The items going across is from {0} to {1}",
                          items.GetLowerBound(0), items.GetUpperBound(0))
        Console.WriteLine("The number of dimensions is {0}", items.Rank)
        For x As Integer = items.GetLowerBound(0) To items.GetUpperBound(0)
            For y As Integer = items.GetLowerBound(1) To items.GetUpperBound(1)
                Console.WriteLine(items(x, y))
                If items(x, y) = 13 Then
                    Console.WriteLine("The condition is in item {0},{1}", x, y)
                End If
            Next
        Next
        For Each itm As Integer In items
            Console.WriteLine(itm)
        Next
    End Sub
End Module
```

Stacks

```
Imports System
Imports System.Collections
```

```
Module Program
    Sub Main(args As String())
        Dim myStack As New Stack
        myStack.Push("A")
        myStack.Push("B")
        myStack.Push("C")
        myStack.Push("D")
        myStack.Push("E")
        Console.WriteLine(myStack.Pop())
        Console.WriteLine(myStack.Peek())
        Console.WriteLine(myStack.Pop())
        myStack.Push("F")
        For Each obj As Object In myStack
            Console.WriteLine("In my stack there is a {0}", obj)
        Next
        If myStack.Contains("B") Then
            Console.WriteLine("Yes, there is a B.")
        End If
        myStack.Clear()
        Console.WriteLine("Here is the contents of my Stack")
        For Each obj As Object In myStack
            Console.WriteLine("In my stack there is a {0}", obj)
        Next
    End Sub
End Module
```

Queues

```
Imports System
Imports System.Collections

Module Program
    Sub Main(args As String())
        Dim myQueue As New Queue
        myQueue.Enqueue("A")
        myQueue.Enqueue("B")
        myQueue.Enqueue("C")
        myQueue.Enqueue("D")
        myQueue.Enqueue("E")
        Console.WriteLine(myQueue.Dequeue())
        Console.WriteLine(myQueue.Peek())
        Console.WriteLine(myQueue.Dequeue())
```

```
        myQueue.Enqueue("F")
        For Each obj As Object In myQueue
            Console.WriteLine("In my Queue there is a {0}", obj)
        Next
        If myQueue.Contains("B") Then
            Console.WriteLine("Yes, there is a B.")
        End If
        myQueue.Clear()
        Console.WriteLine("Here is the contents of my Queue")
        For Each obj As Object In myQueue
            Console.WriteLine("In my Queue there is a {0}", obj)
        Next
    End Sub
End Module
```

Linked Lists

```
Imports System.Collections.Generic

Module Program
    Sub Main(args As String())
        Dim LL As New LinkedList(Of String)
        LL.AddFirst("A")
        LL.AddLast("B")
        LL.AddLast("C")
        LL.AddLast("D")
        LL.AddLast("E")
        LL.RemoveLast()
        LL.Remove(LL.Find("B"))
        LL.AddBefore(LL.Find("C"), "F")
        LL.AddAfter(LL.Find("C"), "G")
        If LL.Contains("B") Then
            Console.WriteLine("The Contains is true")
        Else
            Console.WriteLine("The Contains is false")
        End If
        For Each word As String In LL
            Console.WriteLine(word)
        Next
        LL.Clear()
    End Sub
End Module
```

Bubble Sort

```
' Based on http://rosettacode.org/wiki/Sorting_algorithms/Bubble_sort#Visual_Basic_.NET
Module Program
    Sub Main(args As String())
        Dim testList = {3, 7, 3, 2, 1, -4, 10, 12, 4}
        BubbleSort(testList)
        For Each t In testList
            Console.Write(t & " ")
        Next
    End Sub

    Sub BubbleSort(List As Integer())
        Dim NumberOfItems As Integer = List.Count - 1
        Dim NoMoreSwaps As Boolean
        Dim Temp As Integer
        Do Until NoMoreSwaps = True
            NoMoreSwaps = True
            For Counter = 0 To NumberOfItems - 1
                If List(Counter) > List(Counter + 1) Then
                    NoMoreSwaps = False
                    Temp = List(Counter)
                    List(Counter) = List(Counter + 1)
                    List(Counter + 1) = Temp
                End If
            Next
            NumberOfItems -= 1
        Loop
    End Sub
End Module
```

QuickSort

```
'See https://gist.github.com/ptpt/329260 for the code
```

Understanding web applications

HTML

```
<!DOCTYPE html>

<html>
<head>
```

```
    <meta charset="utf-8" />
    <title>This is my title.</title>
</head>
<body>
    <h1>This is my heading 1.</h1>
    <h2>This is my heading 2.</h2>
    </p><b>HTML</b> is Hypertext Markup Language.</p>
    <i>HTTP</i> is Hypertext Transfer Protocol.</br>
    <u>HTTPS</u> is Hypertext Transfer Protocol Secure.</p>
    This is a link to <a href="http:'www.microsoft.com">Microsoft</a>.
</body>
</html>
```

Adding color

```
    <h1 style="background-color: lightyellow">This is my heading 1.</h1>
    <p style="color:red"><b>HTML</b> is Hypertext Markup Language.</p>
    <p style="font-size:14px;font-family:'Segoe UI', Tahoma, Geneva, Verdana, sans-serif">
```

CSS

```
    <link rel="stylesheet" href="StyleSheet1.css" />

body
{
    color:lawngreen
}
h1 {
    background-color: lightyellow
}
```

JavaScript

```
<head>
    <script>
        function AddFavColor()
        {
            document.getElementById("FavColor").innerHTML = "Your favorite color is " + favoritecolor + "."
        }
    </script>
</head>

<body>
    <p id="FavColor"></p>
```

```
    <script>
        favoritecolor = prompt("Please enter your favorite color")
        AddFavColor()
    </script>
    <noscript>No JavaScript! Please enable for full functionality.</noscript>
</body>
```

ASP.NET

```
Public Class _Default
    Inherits Page

    Protected Sub Page_Load(ByVal sender As Object, ByVal e As EventArgs) Handles Me.Load
        Response.Write("Event: " & System.Reflection.MethodBase.GetCurrentMethod().Name & ".<br/>")
    End Sub

    Private Sub _Default_PreInit(sender As Object, e As EventArgs) Handles Me.PreInit
        Response.Write("Event: " & System.Reflection.MethodBase.GetCurrentMethod().Name & ".<br/>")
    End Sub

    Private Sub _Default_Init(sender As Object, e As EventArgs) Handles Me.Init
        Response.Write("Event: " & System.Reflection.MethodBase.GetCurrentMethod().Name & ".<br/>")
    End Sub

    Private Sub _Default_InitComplete(sender As Object, e As EventArgs) Handles Me.InitComplete
        Response.Write("Event: " & System.Reflection.MethodBase.GetCurrentMethod().Name & ".<br/>")
    End Sub

    Private Sub _Default_PreLoad(sender As Object, e As EventArgs) Handles Me.PreLoad
        Response.Write("Event: " & System.Reflection.MethodBase.GetCurrentMethod().Name & ".<br/>")
    End Sub

    Private Sub _Default_LoadComplete(sender As Object, e As EventArgs) Handles Me.LoadComplete
        Response.Write("Event: " & System.Reflection.MethodBase.GetCurrentMethod().Name & ".<br/>")
    End Sub

    Private Sub _Default_PreRender(sender As Object, e As EventArgs) Handles Me.PreRender
        Response.Write("Event: " & System.Reflection.MethodBase.GetCurrentMethod().Name & ".<br/>")
    End Sub

    Private Sub _Default_PreRenderComplete(sender As Object, e As EventArgs) Handles Me.PreRenderComplete
        Response.Write("Event: " & System.Reflection.MethodBase.GetCurrentMethod().Name & ".<br/>")
```

```
    End Sub

    Private Sub _Default_SaveStateComplete(sender As Object, e As EventArgs) Handles Me.SaveStateComplete
        Response.Write("Event: " & System.Reflection.MethodBase.GetCurrentMethod().Name & ".<br/>")
    End Sub
End Class
```

State Management

```
        Dim myCookie = Request.Cookies.Get("MyParticularCookie")
        Response.Cookies.Add(myCookie)

        ViewState("name") = "Phillip"

        Application("WelcomeToTheSite") = "Welcome back"
        Session("WelcomeToTheSite") = "Welcome back"
```

Web services that will be consumed by client applications

```
Public Class MyWebService
    Inherits System.Web.Services.WebService

    <WebMethod()>
    Public Function HelloWorld() As String
        Return "Hello World"
    End Function

    <WebMethod()>
    Public Function AddUp(a As Integer, b As Integer) As Integer
        Return a + b
    End Function

End Class
```

Accessing web services from a client application

```
        Dim myWeb = New webs.MyWebService
        Response.Write(myWeb.HelloWorld)
        Response.Write(myWeb.AddUp(3, 5))
```

Windows apps

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles AddTogether.Click
        Try
```

```
            MessageBox.Show("The total is " &
                            (CDbl(FirstNumber.Text) + CDbl(SecondNumber.Text)))
        Catch ex As Exception
            MessageBox.Show("I cannot add these numbers together.")
        Finally

        End Try

    End Sub

    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        MsgBox("Load")
    End Sub

    Private Sub Form1_Shown(sender As Object, e As EventArgs) Handles MyBase.Shown
        MsgBox("Shown")
    End Sub

End Class
```

Windows Services

```
Public Class Service1

    Protected Overrides Sub OnStart(ByVal args() As String)
        ' Add code here to start your service. This method should set things
        ' in motion so your service can do its work.
        EventLog.WriteEntry("I've started.")
    End Sub

    Protected Overrides Sub OnStop()
        ' Add code here to perform any tear-down necessary to stop your service.
        EventLog.WriteEntry("I've finished.")
    End Sub

End Class
```

Understanding databases

Inserting, Updating and Deleting data

```
INSERT [dbo].[tblEmployee]([EmployeeFirstName], [EmployeeLastName])
VALUES ('Jane','Smith'), ('Fred','Bloggs')
```

```
UPDATE [dbo].[tblEmployee]
SET [EmployeeLastName] = 'Smyth'
WHERE [EmployeeLastName] = 'Smith'

DELETE [dbo].[tblEmployee]
WHERE [EmployeeLastName] like 'B%'

select * from [dbo].[tblEmployee]
```

Stored Procedures and Views

```
select *
from [dbo].[MyView]
where EmployeeLastName like 'S%'

exec [dbo].[MyProcedure]
```

Accessing SQL Server in Visual Studio

```
Imports System.Data.SqlClient
    Private Sub btnSQLServer_Click(sender As Object, e As EventArgs) Handles btnSQLServer.Click
        Dim builder As New SqlConnectionStringBuilder
        builder.DataSource = "SILENTAND\SQLEXPRESS"
        builder.InitialCatalog = "98-361"
        builder.IntegratedSecurity = True
        Dim myconnection = New SqlConnection(builder.ConnectionString)
        myconnection.Open()
        Dim mycommand = New SqlCommand("select * from [dbo].[tblEmployee]", myconnection)
        Dim myreader = mycommand.ExecuteReader
        While myreader.Read
            MessageBox.Show(myreader.GetString(0) & " " & myreader.GetString(1))
        End While
    End Sub
```

Reading Flat Files in Visual Studio

```
Imports System.IO

        Dim myStreamReader = New StreamReader("C:\inetpub7\flatfile.txt")
        Dim myStreamReaderInput = myStreamReader.ReadLine()
        While myStreamReaderInput <> Nothing
            MessageBox.Show(myStreamReaderInput)
            myStreamReaderInput = myStreamReader.ReadLine()
```

```
        End While
        myStreamReader.Close()
```

Reading XML Files in Visual Studio

```
select * from [dbo].[tblEmployee] as Employee
FOR XML AUTO, ELEMENTS, ROOT('Employees')
```

```
        Dim myStreamReader = New StreamReader("c:\inetpub7\XMLfile.xml")
        Dim mySettings = New XmlReaderSettings
        Dim myReader = XmlReader.Create(myStreamReader, mySettings)
        While myReader.Read
            Select Case myReader.NodeType
                Case XmlNodeType.Element
                    MessageBox.Show(myReader.Name)
                Case XmlNodeType.Text
                    MessageBox.Show(myReader.Value)
                Case XmlNodeType.EndElement
                    MessageBox.Show("   - End of " & myReader.Name)
            End Select

        End While
```

Reading In-memory objects

```
using System.IO
using System.Xml
```

```
        Dim myDataSet = New DataSet
        myDataSet.ReadXml("c:\inetpub7\XMLfile.xml")
        For Each myTable As DataTable In myDataSet.Tables
            MessageBox.Show("New table " & myTable.TableName)
            For Each myRow As DataRow In myTable.Rows
                For Each myColumn As DataColumn In myTable.Columns
                    MessageBox.Show(myColumn.ColumnName & " " & myRow(myColumn))
                Next
            Next
        Next
```