## Comments

<table>
<thead>
<tr>
<th>VB.NET</th>
<th>C#</th>
</tr>
</thead>
<tbody>
<tr>
<td>'Single line only Rem Single line only</td>
<td>// Single line /* Multiple line <em>/ /</em> XML comments on single line /** XML comments on multiple lines */</td>
</tr>
</tbody>
</table>

## Program Structure

<table>
<thead>
<tr>
<th>VB.NET</th>
<th>C#</th>
</tr>
</thead>
<tbody>
<tr>
<td>Imports System Namespace MyNameSpace Class HelloWorld 'Entry point which delegates to C-style main Private Function Public Overloads Shared Sub Main() Main(System.Environment.GetCommandLineArgs()) End Sub Overloads Shared Sub Main(args() As String) System.Console.WriteLine(&quot;Hello World&quot;) End Sub 'Main End Class 'HelloWorld End Namespace 'MyNameSpace</td>
<td>using System Namespace MyNameSpace { class HelloWorld { static void Main(string[] args) { System.Console.WriteLine(&quot;Hello World&quot;) } } }</td>
</tr>
</tbody>
</table>
## Data Types

### VB.NET

- 'Value Types
  - Boolean
  - Byte
  - Char (example: "A")
  - Short, Integer, Long
  - Single, Double
  - Decimal
  - Date

- 'Reference Types
  - Object
  - String

### C#

- //Value Types
  - bool
  - byte, sbyte
  - char (example: 'A')
  - short, ushort, int, uint, long, ulong
  - float, double
  - decimal
  - DateTime

- //Reference Types
  - object
  - string

### Type conversion

- Dim d As Single = 3.5
- Dim i As Integer = CType (d, Integer)
- i = CInt (d)
  
### Constants

### VB.NET

- Const MAX_AUTHORS As Integer = 25
- ReadOnly MIN_RANK As Single = 5.00

### C#

- const int MAX_AUTHORS = 25;
- readonly float MIN_RANKING = 5.00;
### Enumerations

#### VB.NET

<table>
<thead>
<tr>
<th>Enum Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start</td>
</tr>
<tr>
<td>Stop</td>
</tr>
<tr>
<td>Rewind</td>
</tr>
<tr>
<td>Forward</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Enum Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flunk = 50</td>
</tr>
<tr>
<td>Pass = 70</td>
</tr>
<tr>
<td>Excel = 90</td>
</tr>
</tbody>
</table>

```vbnet
Dim a As Action = Action.Stop
If a <> Action.Start Then
    'Prints "Stop is 1"
    System.Console.WriteLine(a.ToString() & " is " & a)

'Prints 70
System.Console.WriteLine(Status.Pass)
'Prints Pass
```

#### C#

```csharp
enum Action {Start, Stop, Rewind, Forward};
enum Status {Flunk = 50, Pass = 70, Excel = 90};

Action a = Action.Stop;
if (a != Action.Start)
    //Prints "Stop is 1"
    System.Console.WriteLine(a + " is " + (int)a);

// Prints 70
System.Console.WriteLine((int)Status.Pass);
// Prints Pass
System.Console.WriteLine(Status.Pass);
```

#### VB.NET

<table>
<thead>
<tr>
<th>Enum Weekdays</th>
</tr>
</thead>
<tbody>
<tr>
<td>Saturday</td>
</tr>
<tr>
<td>Sunday</td>
</tr>
<tr>
<td>Monday</td>
</tr>
<tr>
<td>Tuesday</td>
</tr>
<tr>
<td>Wednesday</td>
</tr>
<tr>
<td>Thursday</td>
</tr>
<tr>
<td>Friday</td>
</tr>
</tbody>
</table>

```vbnet
Enum Weekdays
    Saturday
    Sunday
    Monday
    Tuesday
    Wednesday
    Thursday
    Friday
End Enum 'Weekdays
```

#### C#

```csharp
enum Weekdays
{
    Saturday, Sunday, Monday, Tuesday,
    Wednesday, Thursday, Friday
}
```
## Operators

### VB.NET

<table>
<thead>
<tr>
<th>Operators</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>'Comparison</td>
<td>=  &lt;  &gt;  &lt;=  &gt;=  &lt;&gt;</td>
</tr>
<tr>
<td>'Arithmetic</td>
<td>+  -  *  /  \  (integer division)  ^  (raise to a power)</td>
</tr>
<tr>
<td>'Assignment</td>
<td>=  +=  -=  *=  /=  =  ^=  &lt;&lt;=  &gt;&gt;=  &amp;=</td>
</tr>
<tr>
<td>'Bitwise</td>
<td>And  AndAlso  Or  OrElse  Not  &lt;&lt;  &gt;&gt;</td>
</tr>
<tr>
<td>'Logical</td>
<td>And  AndAlso  Or  OrElse  Not</td>
</tr>
<tr>
<td>'String Concatenation</td>
<td>&amp;</td>
</tr>
</tbody>
</table>

### C#

<table>
<thead>
<tr>
<th>Operators</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>//Comparison</td>
<td>==  &lt;  &gt;  &lt;=  &gt;=  !=</td>
</tr>
<tr>
<td>//Arithmetic</td>
<td>+  -  *  /  %  (mod)  /  (integer division if both operands are ints) Math.Pow(x, y)</td>
</tr>
<tr>
<td>//Assignment</td>
<td>=  +=  -=  *=  /=  %=  &amp;=</td>
</tr>
<tr>
<td>//Bitwise</td>
<td>&amp;</td>
</tr>
<tr>
<td>//Logical</td>
<td>&amp;</td>
</tr>
<tr>
<td>//String Concatenation</td>
<td>+</td>
</tr>
<tr>
<td><strong>VB.NET</strong></td>
<td><strong>C#</strong></td>
</tr>
<tr>
<td>------------</td>
<td>--------</td>
</tr>
<tr>
<td>greeting = IIf(age &lt; 20, &quot;What's up?&quot;, &quot;Hello&quot;)</td>
<td>greeting = age &lt; 20 ? &quot;What's up?&quot; : &quot;Hello&quot;;</td>
</tr>
</tbody>
</table>

'One line doesn't require "End If", no "Else"  
If language = "VB.NET" Then langType = "verbose" |

'Use: to put two commands on same line  
If x <> 100 And y < 5 Then x *= 5 : y *= 2 |

'Preferred  
If x <> 100 And y < 5 Then  
x *= 5  
y *= 2  
End If |

'or to break up any long single command use _  
If henYouHaveAReally < longLine And _  
itNeedsToBeBrokenInto2 > Lines _ _  
UseTheUnderscore(charToBreakItUp) |

If x > 5 Then  
x *= y  
ElseIf x = 5 Then  
x += y  
ElseIf x < 10 Then  
x -= y  
Else  
x /= y  
End If |

'Must be a primitive data type  
Select Case color  
Case "black", "red"  
r += 1  
Case "blue"  
b += 1  
Case "green"  
g += 1  
Case Else  
other += 1  
End Select |

//Must be integer or string  
switch (color)  
{  
case "black":  
r++;  
case "red":  
r++;  
break;  
case "blue":  
break;  
case "green":  
g++;  
break;  
default:  
other++;  
break;  
}
<table>
<thead>
<tr>
<th>VB.NET</th>
<th>C#</th>
</tr>
</thead>
<tbody>
<tr>
<td>'Pre-test Loops: While c &lt; 10 c += 1 End While Do Until c = 10 c += 1 Loop</td>
<td>//Pre-test Loops: while (i &lt; 10) i++; for (i = 2; i &lt;= 10; i += 2) System.Console.WriteLine(i);</td>
</tr>
<tr>
<td>'Post-test Loop: Do While c &lt; 10 c += 1 Loop</td>
<td>//Post-test Loop: do i++; while (i &lt; 10);</td>
</tr>
<tr>
<td>For c = 2 To 10 Step 2 System.Console.WriteLine(c) Next</td>
<td>// Array or collection looping string[] names = {&quot;Steven&quot;, &quot;SuOk&quot;, &quot;Sarah&quot;); foreach (string s in names) System.Console.WriteLine(s);</td>
</tr>
<tr>
<td>'Array or collection looping Dim names As String() = (&quot;Steven&quot;, &quot;SuOk&quot;, &quot;Sarah&quot;) For Each s As String In names System.Console.WriteLine(s) Next</td>
<td></td>
</tr>
</tbody>
</table>
### Arrays

**VB.NET**

```vbnet
Dim nums() As Integer = {1, 2, 3}
For i As Integer = 0 To nums.Length - 1
    Console.WriteLine(nums(i))
Next

'4 is the index of the last element, so it holds 5 elements
Dim names(4) As String
names(0) = "Steven"
'Throws System.OutOfRange
names(5) = "Sarah"

'Resize the array, keeping the existing values (Preserve is optional)
ReDim Preserve names(6)
```

```vbnet
Dim twoD(rows-1, cols-1) As Single
twoD(2, 0) = 4.5
```

```vbnet
Dim jagged()() As Integer = { New Integer(4) {}, New Integer(1) {}, New Integer(2) {} }
jagged(0)(4) = 5
```

**C#**

```csharp
int[] nums = {1, 2, 3};
for (int i = 0; i < nums.Length; i++)
    Console.WriteLine(nums[i]);

// 5 is the size of the array
string[] names = new string[5];
names[0] = "Steven";
// Throws System.OutOfRange
names[5] = "Sarah"

// C# can't dynamically resize an array. //Just copy into new array.
string[] names2 = new string[7];
// or names.CopyTo(names2, 0);
Array.Copy(names, names2, names.Length);
```

```csharp
float[,] twoD = new float[rows, cols];
twoD[2,0] = 4.5;
```

```csharp
int[][] jagged = new int[3][] {
    new int[5], new int[2], new int[3] 
};
jagged[0][4] = 5;
```
### VB.NET

<table>
<thead>
<tr>
<th>Functions</th>
</tr>
</thead>
</table>
| 'Pass by value (in, default), reference
'(in/out), and reference (out)
Sub TestFunc(ByVal x As Integer, ByRef y As Integer,
ByRef z As Integer)
  x += 1
  y += 1
  z = 5
End Sub |

<table>
<thead>
<tr>
<th>C#</th>
</tr>
</thead>
</table>
| // Pass by value (in, default), reference
//(in/out), and reference (out)
void TestFunc(int x, ref int y, out int z)
{
  x++;
  y++;
  z = 5;
} |

| 'c set to zero by default
Dim a = 1, b = 1, c As Integer
TestFunc(a, b, c)
System.Console.WriteLine("{0} {1} {2}", a, b, c) '1 2 5 |

| int a = 1, b = 1, c; // c doesn't need initializing
TestFunc(a, ref b, out c);
System.Console.WriteLine("{0} {1} {2}", a, b, c); // 1 2 5 |

| 'Accept variable number of arguments
Function Sum(ByVal ParamArray nums As Integer()) As Integer
  Sum = 0
  For Each i As Integer In nums
    Sum += i
  Next
End Function 'Or use a Return statement like C#
Dim total As Integer = Sum(4, 3, 2, 1) 'returns 10 |

| int total = Sum(4, 3, 2, 1); // returns 10 |

| 'Optional parameters must be listed last
'and must have a default value
Sub SayHello(ByVal name As String,
Optional ByVal prefix As String = "")
  System.Console.WriteLine("Greetings, " & prefix & " " & name)
End Sub |

| /* C# doesn't support optional arguments/parameters.
Just create two different versions of the same function. */
void SayHello(string name, string prefix) {
  System.Console.WriteLine("Greetings, " + prefix + " " + name);
}
void SayHello(string name) {
  SayHello(name, ""); |

SayHello("Steven", "Dr.")
SayHello("SuOk")
## Exception Handling

### VB.NET

```vbnet
Class Withfinally
    Public Shared Sub Main()
        Try
            Dim x As Integer = 5
            Dim y As Integer = 0
            Dim z As Integer = x / y
            Console.WriteLine(z)
        Catch e As DivideByZeroException
            System.Console.WriteLine("Error occurred")
        Finally
            System.Console.WriteLine("Thank you")
        End Try
    End Sub 'Main
End Class 'Withfinally
```

### C#

```csharp
class Withfinally
{
    public static void Main()
    {
        try
        {
            int x = 5;
            int y = 0;
            int z = x/y;
            Console.WriteLine(z);
        }
        catch(DivideByZeroException e)
        {
            System.Console.WriteLine("Error occurred");
        }
        finally
        {
            System.Console.WriteLine("Thank you");
        }
    }
}
```

## Namespaces

### VB.NET

```vbnet
Namespace ASPAlliance.DotNet.Community
    ...
End Namespace

'or

namespace ASPAlliance.DotNet.Community {
    ...
}
```

### C#

```csharp
namespace ASPAlliance.DotNet.Community {
    ...
}
```

```
namespace ASPAlliance {
    namespace DotNet {
        namespace Community {
        ...}
    }
}
```

```csharp
using ASPAlliance.DotNet.Community;
```
## Classes / Interfaces

### VB.NET

<table>
<thead>
<tr>
<th>'Accessibility keywords</th>
<th>Public</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Private</td>
</tr>
<tr>
<td></td>
<td>Friend</td>
</tr>
<tr>
<td></td>
<td>Protected</td>
</tr>
<tr>
<td></td>
<td>Protected Friend</td>
</tr>
<tr>
<td></td>
<td>Shared</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>'Inheritance</th>
<th>Class Articles</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Inherits Authors</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>End Class</td>
</tr>
</tbody>
</table>

**Imports System**

**Interface IArticle**

- Sub Show()

**End Interface 'IArticle**

**Class IAuthor**

- Implements IArticle

- Public Sub Show()

  System.Console.WriteLine("Show() method Implemented");

**End Sub 'Show**

**'Entry point which delegates to C-style main**

**Private Function**

- Public Overloads Shared Sub Main()

  Main(System.Environment.GetCommandLineArgs())

**End Sub**

**Overloads Public Shared Sub Main(args() As String)**

- Dim author As New IAuthor()

  author.Show()

**End Sub 'Main**

**End Class 'IAuthor**

### C#

<table>
<thead>
<tr>
<th>//Accessibility keywords</th>
<th>public</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>private</td>
</tr>
<tr>
<td></td>
<td>internal</td>
</tr>
<tr>
<td></td>
<td>protected</td>
</tr>
<tr>
<td></td>
<td>protected internal</td>
</tr>
<tr>
<td></td>
<td>static</td>
</tr>
</tbody>
</table>

| //Inheritance            | class Articles: Authors {
|-------------------------| ...                    |
|                         | }                      |

**using System;**

**interface IArticle**

- void Show();

**class IAuthor:IArticle**

- public void Show()

  System.Console.WriteLine("Show() method Implemented");

- public static void Main(string[] args)

  IAuthor author = new IAuthor();
  author.Show();

**End Sub**
### Constructors / Destructors

<table>
<thead>
<tr>
<th>VB.NET</th>
<th>C#</th>
</tr>
</thead>
</table>
| Class TopAuthor  
  Private _topAuthor As Integer  
  Public Sub New()  
    _topAuthor = 0  
  End Sub  
  Public Sub New(ByVal topAuthor As Integer)  
    Me._topAuthor = topAuthor  
  End Sub  
  Protected Overrides Sub Finalize()  
    'Destructor code to free unmanaged resources  
    MyBase.Finalize()  
  End Sub | class TopAuthor {  
  private int _topAuthor;  
  public TopAuthor() {  
    _topAuthor = 0;  
  }  
  public TopAuthor(int topAuthor) {  
    this._topAuthor= topAuthor  
  }  
  ~TopAuthor() {  
    // Destructor code to free unmanaged resources.  
    // Implicitly creates a Finalize method  
  } |

### Objects

<table>
<thead>
<tr>
<th>VB.NET</th>
<th>C#</th>
</tr>
</thead>
</table>
| Dim author As TopAuthor = New TopAuthor  
  With author  
    .Name = "Steven"  
    .AuthorRanking = 3  
  End With  
  author.Rank("Scott")  
  author.Demote() 'Calling Shared method  
  'or  
  TopAuthor.Rank()  
  Dim author2 As TopAuthor = author 'Both refer to same object  
  author2.Name = "Joe"  
  System.Console.WriteLine(author2.Name) 'Prints Joe  
  author = Nothing 'Free the object  
  If author Is Nothing Then _  
    author = New TopAuthor  
  End If  
  Dim obj As Object = New TopAuthor  
  If TypeOf obj Is TopAuthor Then _  
    System.Console.WriteLine("Is a TopAuthor object.") | TopAuthor author = new TopAuthor();  
  //No "With" construct  
  author.Name = "Steven";  
  author.AuthorRanking = 3;  
  TopAuthor author2 = author 'Both refer to same object  
  author2.Name = "Joe";  
  System.Console.WriteLine(author2.Name) //Prints Joe  
  author = null //Free the object  
  if (author == null)  
    author = new TopAuthor();  
  Object obj = new TopAuthor();  
  if (obj is TopAuthor)  
    System.Console.WriteLine("Is a TopAuthor object."); |
## Structs

### VB.NET

Structure AuthorRecord
Public name As String
Public rank As Single

Public Sub New(ByVal name As String, ByVal rank As Single)
    Me.name = name
    Me.rank = rank
End Sub
End Structure

Dim author As AuthorRecord = New AuthorRecord("Steven", 8.8)
Dim author2 As AuthorRecord = author
author2.name = "Scott"
System.Console.WriteLine(author.name) 'Prints Steven
System.Console.WriteLine(author2.name) 'Prints Scott

### C#

```csharp
struct AuthorRecord {
    public string name;
    public float rank;

    public AuthorRecord(string name, float rank) {
        this.name = name;
        this.rank = rank;
    }
}
```

AuthorRecord author = new AuthorRecord("Steven", 8.8);
AuthorRecord author2 = author
author.name = "Scott";
System.Console.WriteLine(author.name); //Prints Steven
System.Console.WriteLine(author2.name); //Prints Scott
### Properties

<table>
<thead>
<tr>
<th>VB.NET</th>
<th>C#</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Private _size As Integer</strong></td>
<td><strong>private int _size;</strong></td>
</tr>
<tr>
<td><strong>Public Property Size() As Integer</strong></td>
<td><strong>public int Size {</strong></td>
</tr>
<tr>
<td>Get</td>
<td>get {</td>
</tr>
<tr>
<td>Return _size</td>
<td>return _size;</td>
</tr>
<tr>
<td>End Get</td>
<td>}</td>
</tr>
<tr>
<td>Set (ByVal Value As Integer)</td>
<td>set {</td>
</tr>
<tr>
<td>If Value &lt; 0 Then</td>
<td>if (value &lt; 0)</td>
</tr>
<tr>
<td>_size = 0</td>
<td>_size = 0;</td>
</tr>
<tr>
<td>Else</td>
<td>else</td>
</tr>
<tr>
<td>_size = Value</td>
<td>_size = value;</td>
</tr>
<tr>
<td>End If</td>
<td>}</td>
</tr>
<tr>
<td>End Set</td>
<td>}</td>
</tr>
<tr>
<td>End Property</td>
<td>}</td>
</tr>
</tbody>
</table>

```vbnet
foo.Size += 1
```

```csharp
using System;
class Date {
    public int Day {
        get {
            return day;
        } 
        set {
            day = value;
        }
    }
    int day;
    public int Month {
        get {
            return month;
        } 
        set {
            month = value;
        }
    }
    int month;
    public int Year {
        get {
            return year;
        } 
        set {
            year = value;
        }
    }
    int year;
    public bool IsLeapYear(int year) {
        return year % 4 == 0 ? true : false;
    }
```
Delegates / Events

**VB.NET**

Delegate Sub MsgArrivedEventHandler(ByVal message As String)

Event MsgArrivedEvent As MsgArrivedEventHandler

'or to define an event which declares a 'delegate implicitly
Event MsgArrivedEvent(ByVal message As String)

AddHandler MsgArrivedEvent, AddressOf My_MsgArrivedCallback
My_MsgArrivedCallback

RaiseEvent MsgArrivedEvent("Test message")

RemoveHandler MsgArrivedEvent, AddressOf My_MsgArrivedCallback
My_MsgArrivedCallback

Imports System.Windows.Forms

'WithEvents can't be used on local variable
Dim WithEvents MyButton As Button
MyButton = New Button

Private Sub MyButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyButton.Click
    MessageBox.Show(Me, "Button was clicked", "Info", MessageBoxButtons.OK, MessageBoxIcon.Information)
End Sub

**C#**

delegate void MsgArrivedEventHandler(string message);

event MsgArrivedEventHandler

//Delegates must be used with events in C#

MsgArrivedEvent += new MsgArrivedEventHandler(My_MsgArrivedEventCallback);

//Throws exception if obj is null
MsgArrivedEvent("Test message");

MsgArrivedEvent -= new MsgArrivedEventHandler(My_MsgArrivedEventCallback);

using System.Windows.Forms;

Button MyButton = new Button();
MyButton.Click += new System.EventHandler(MyButton_Click);

private void MyButton_Click(object sender, System.EventArgs e) {
    MessageBox.Show(this, "Button was clicked", "Info", MessageBoxButtons.OK,
    MessageBoxIcon.Information);
}
## Console I/O

### VB.NET

'Special character constants  
vbCrLf, vbCr, vbLf, vbNewLine  
vbNullString  
vbTab  
vbBack  
vbFormFeed  
vbVerticalTab  
""  
Chr(65) 'Returns 'A'

```vbnet
System.Console.Write("What's your name? ")  
Dim name As String = System.Console.ReadLine()  
Dim age As Integer = Val(System.Console.ReadLine())  
System.Console.WriteLine("{0} is {1} years old.", name, age)  
'or  
System.Console.WriteLine(name & " is " & age & " years old.")
```

```vbnet
Dim c As Integer  
c = System.Console.Read() 'Read single char  
System.Console.WriteLine(c) 'Prints 65 if user enters "A"
```

### C#

//Escape sequences  
\n, \r  
\t  
\\  
Convert.ToChar(65) //Returns 'A' - equivalent to Chr(num) in VB  
// or  
(char) 65

```csharp
System.Console.Write("What's your name? ");  
string name = System.Console.ReadLine();  
System.Console.Write("How old are you? ");  
int age = Convert.ToInt32(System.Console.ReadLine());  
System.Console.WriteLine("{0} is {1} years old.", name, age);  
'or  
System.Console.WriteLine(name + " is " + age + " years old.");
```

```csharp
int c = System.Console.Read(); //Read single char  
System.Console.WriteLine(c); //Prints 65 if user enters "A"
```
### File I/O

#### VB.NET

```vbnet
Imports System.IO

'Write out to text file
Dim writer As StreamWriter = File.CreateText("c:\myfile.txt")
writer.WriteLine("Out to file.")
writer.Close()

'REad all lines from text file
Dim reader As StreamReader = File.OpenText("c:\myfile.txt")
Dim line As String = reader.ReadLine()
While Not line Is Nothing
    Console.WriteLine(line)
    line = reader.ReadLine()
End While
reader.Close()

'Write out to binary file
Dim str As String = "Text data"
Dim num As Integer = 123
Dim binWriter As New BinaryWriter(File.OpenWrite("c:\myfile.dat"))
binWriter.Write(str)
binWriter.Write(num)
binWriter.Close()

'REad from binary file
Dim binReader As New BinaryReader(File.OpenRead("c:\myfile.dat"))
str = binReader.ReadString()
num = binReader.ReadInt32()
binReader.Close()
```

#### C#

```csharp
using System.IO;

//Write out to text file
StreamWriter writer = File.CreateText("c:\myfile.txt");
writer.WriteLine("Out to file.");
writer.Close();

//Read all lines from text file
StreamReader reader = File.OpenText("c:\myfile.txt");
string line = reader.ReadLine();
while (line != null)
{
    Console.WriteLine(line);
    line = reader.ReadLine();
}
reader.Close();

//Write out to binary file
string str = "Text data";
int num = 123;
BinaryWriter binWriter = new BinaryWriter(File.OpenWrite("c:\myfile.dat"));
binWriter.Write(str);
binWriter.Write(num);
binWriter.Close();

//Read from binary file
BinaryReader binReader = new BinaryReader(File.OpenRead("c:\myfile.dat"));
str = binReader.ReadString();
num = binReader.ReadInt32();
binReader.Close();
```