

Comments

```
// Single line
/* Multiple
line */
/// XML comments on
single line
/* XML comments on
multiple lines /
```

Enumerations

```
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(Statu s.P -
ass);
enum Weekdays{
    Saturday, Sunday,
Monday, Tuesday,
Wednesday, Thursday,
Friday
}
```

Delegates / Events

```
delegate void
MsgArrivedEventHandler(string
message);
```

Delegates / Events (cont)

```
> event MsgArrivedEvent-
Handler MsgArrivedEvent;

//Delegates must be used with
events in C#

MsgArrivedEvent += new
MsgArrivedEventHandler
(My_MsgArrivedEventCallb-
ack);
//Throws exception if obj is null
MsgArrivedEvent("Test messag-
e");
MsgArrivedEvent -= new
MsgArrivedEventHandler
(My_MsgArrivedEventCallb-
ack);

using System.Windows.Forms;

Button MyButton = new Button();
MyButton.Click += new
System.EventHandler(MyButto-
n_Click);
private void MyButton_Click(-
object sender, System.Ev-
entArgs e) {
    MessageBox.Show(this, "-
Button was clicked", "Info",
    MessageBoxButtons.OK,
    MessageBoxIcon.Information);
}
```

Loops

```
//Pre-test Loops: while
(i < 10)
    i++;
for (i = 2; i <= 10; i
+= 2)
    System.Console.Wri-
teLine(i);
//Post -test Loop:
do
    i++;
while (i < 10);
// Array or collection
looping
string[] names = {"St -
eve n", " SuO k", " -
Sar ah"};
foreach (string s in
names)
    System.Console.W-
riteLine(s);
```

Objects

```
TopAuthor author = new
TopAuthor();

//No " With" construct
author.Name = " Ste -
ven ";
author.Au tho rRa nking
= 3;

author.Ra nk( " Sco -
tt");
TopAut hor.De mote()
//Calling static method

TopAuthor author2 =
author //Both refer to
same object
author 2.Name = " -
Joe ";
```

Objects (cont)

```
> System.Console.WriteLine(au-
thor2.Name) //Prints Joe

author = null //Free the object

if (author == null)
    author = new TopAuthor();

Object obj = new TopAuthor();
if (obj is TopAuthor)
    SystConsole.WriteLine("Is a
TopAuthor object.");
```

Namespaces

```
namespace
ASPAlliance.DotNet.Communi-
ty
{
    ...
}

// or

namespace ASPAll iance {
    nam espace DotNet {
        nam espace Communi-
        ty
        {
            ...
        }
    }
}
using ASPAll ian ce.D ot -
Net.Co mmu nity;
```

Program Structure

```
using System
Namespace MyName Space{
    class HelloWorld {
```

Program Structure (cont)

```
> static void Main(string[]
args) {
    System.Console.WriteLine-
("Hello World")
}
}
```

Console I/O

```
//Escape sequences
\n, \r
\t
\\
\

Conver t.T oCh ar(65)
//Returns 'A' -
equivalent to Chr(num)
in VB
// or
(char) 65

System.Co nso le.W ri -
te( " What's your name?
");
string name =
SYstem.Co nso le.R ea -
dLi ne();
System.Co nso le.W ri -
te( "How old are you?
");
int age = Conver t.T -
oIn t32 (Sy ste m.C -
ons ole.Re adL ine());
System.Co nso le.W ri -
teL ine ("{0} is {1}
years old.", name, age);
//or
System.Co nso le.W ri -
teL ine (name + " is " +
age + " years old.");
```

Console I/O (cont)

```
> int c = System.Console.R-
ead(); //Read single char
System.Console.WriteLine(c);
//Prints 65 if user enters "A"
```

Operators

```
//Comparison
== < > <= >= !=

//Arit hmetic
+ - * /
% (mod)
/ (integer division if
both operands are ints)
Math.P ow(x, y)

//Assi gnment
= += -= *= /= %= &= |=
^= <<= >>= ++ --

//Bitwise
& | ^ ~ << >>

//Logical
&& || !

//String Concat enation
+
```

Structs

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

    public Author Rec -
ord (string name, float
rank) {
        thi s.name =
name;
        thi s.rank =
rank;
    }
}
```

Structs (cont)

```
> }
AuthorRecord author = new
AuthorRecord("Steven", 8.8);
AuthorRecord author2 = author

author.name = "Scott";
SystemConsole.WriteLine(aut-
hor.name); //Prints Steven
System.Console.WriteLine(au-
thor2.name); //Prints Scott
```

Functions

```
// Pass by value (in,
default), reference
//(in/ out), and
reference (out)
void TestFu nc(int x,
ref int y, out int z) {
    x++;
    y++;
    z = 5;
}

int a = 1, b = 1, c; //
c doesn't need initia -
lizing
TestFu nc(a, ref b, out
c);
System.Co nso le.W ri -
teL ine ("{0} {1} {2}",
a, b, c); // 1 2 5

// Accept variable
number of arguments
```

Functions (cont)

```
> int Sum(params int[] nums) {
    int sum = 0;
    foreach (int i in nums)
        sum += i;
    return sum;
}

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

/* 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("Gr-
eetings, " + prefix + " " + name);
}

void SayHello(string name) {
    SayHello(name, "");
}
```

File I/O

```
using System.IO;
//Write out to text file
Stream Writer writer =
File.C rea teText
("c: \m yfi le.t -
xt ");
writer.Wr ite Lin e("Out
to file.");
writer.Cl ose();
```

File I/O (cont)	File I/O (cont)	Arrays	Data Types
<pre>> //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();</pre>	<pre>> binReader.Close(); Classes / Interfaces //Accessibility keywords public private internal protected protected internal static //Inheritance class Articles: Authors { ... } using System; interface IArticle{ void Show(); } class IAuthor:I Article{ public void Show() { System.Console.WriteLine("Show() method Implemented"); } public static void Main(string[] args) { IAuthor author = new IAuthor(); author.Show(); } }</pre>	<pre>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.Index OutOfRangeException 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); float[,] twoD = new float[rows, cols]; twoD[2,0] = 4.5; int[][] jagged = new int[3][] { new int[5], new int[2], new int[3] }; jagged[0][4] = 5;</pre>	<pre>//Value Types bool byte, sbyte char (example: 'A') short, ushort, int, uint, long, ulong float, double decimal DateTime //Reference Types object string int x; Console.WriteLine(x.GetType()) Console.WriteLine(typeof(int)) //Type conversion float d = 3.5; int i = (int) d</pre>
			Constructors / Destructors <pre>class TopAuthor { private int _topAuthor; public TopAuthor() { _topAuthor = 0; } public TopAuthor(int topAuthor) { this._topAuthor = topAuthor } ~TopAuthor() { // Destructor code to free unmanaged resources. } }</pre>

Constructors / Destructors (cont)

```
> // Implicitly creates a
Finalize method
}
}
```

Exception Handling

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

Properties

```
private int _size;

public int Size {
    get {
        return _size;
    }
    set {
        if (value < 0)
            _size = 0;
    }
}
```

Properties (cont)

```
> else
    _size = value;
}
```

```
foo.Size++;
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;
```

Properties (cont)

```
> public bool IsLeapYear(int
year) {
    return year%4== 0 ? true:
false;
}
public void SetDate (int day,
int month, int year) {
    this.day = day;
    this.month = month;
    this.year = year;
}
}
```

Choices

```
greeting = age < 20 ?
"What's up?" : "Hello";
if (x != 100 && y < 5){
    // Multiple
statements must be
enclosed in {}
    x *= 5;
    y *= 2;
}
if (x > 5)
    x *= y;
else if (x == 5)
    x += y;
else if (x < 10)
    x -= y;
else
    x /= y;
//Must be integer or
string
switch (color){
    case "black":
```

Choices (cont)

```
> case "red": r++;
    break;
case "blue"
    break;
case "green": g++;
    break;
default: other++;
    break;
}
```

Constants

```
const int MAX_AUTHORS =
25;
readonly float MIN_RATING = 5.00;
```