C# Reference Sheet

Basics

Namespace
using Namespace;

Data Types
byte,sbyte,int,uint,short,ushort,long,ulong,float,double,decimal,bool,char,string, object

Variable Declaration
public | protected internal | protected | internal | private <type> As <variable_name>

Type Declaration
public | internal | private <variable><suffix>

Suffixes
f -float, l,L - long, No double suffix, U,u - unsigned

Arrays
<type>[] <name> = new <type>[ArraySize];

Initialize Array
<type>[] <name> = new <type>[ArraySize] {<value1>, <value2>, ... , <valueN>};

Change Size of Array
<type>[] <name> = new <type>[ArraySize];
Array.Resize<type>(ref <name>, <size>);

Comments
//Comment text
Multi-line comments
/ This is commented */*

XML Comments
Press the / (forward slash) key 3 times.

Line Continuation
string strtext = @"To break a long string across multiple lines,
end the string, add the line continuation character
and continue the string on the next line.";

Arithmetic Operators
*+ (Addition), - (Subtraction), * (Multiplication), / (Division), % (Modulus)*

String Concatenation
+

Relational Operators
< (Less Than), <= (Less Than or Equal To), > (Greater Than), >= (Greater Than or Equal To), == (Equal To),! = (Not Equal To), is, as

Logical Operators
& (And), | (Or), ^ (Xor),&& (AndAlso), || (OrElse)

Assignment Operators
*= (Equals), += (Addition), -= (Subtraction), *= (Multiplication), /= (Division), %= (Modulus), &= (And),|= (OR), ^= (Exclusive OR), <<= (Left Shift), >>= (Right Shift), ??*

String Manipulation
.Substring(<start>,[<length>])
.Trim() <trims from beginning & end of string>
.TrimEnd([<char array>])
.TrimStart([char array])
.ToLower() <to lower case>
.ToUpper() <to upper case>
.Replace(<find>,<replace>)
.Equals(<expression>) <6 available overloads>
.Contains(<string>)
.Join(<seperator>,<value>,[<count>])
.Compare(<string1>,<string2>,[<ignore case>]) <7 overloads available>
.Copy(<string>)

Error Handling
```
try
{
    //<statements that may cause an error>;
}
catch(Exception ex)
{
    //<statements to use when an error occurs>;
}
finally
{
    //<statements to use no matter what happens>
}
```

If Else
```
if(expression)
{
    <statement 1>;
}
else
{
    <statement 2>;
}
```

C# version of IIF()
variable == ?true:false;

For Loop
```
for(statement)
{
   <statement>;
}
```

For Each Loop
```
foreach(<variable> In <object>)
{
  <statements>;
  [break];
  [continue];
}
```

While Loop
```
while(<expression>)
{
   <statement>
}
```

Do-While Loop
```
do
{
    <statement>;
} while <expression>;
```

Select Case Statement
```
switch(<expression>)
{
    case <literal or type>:
        <statement>;
        <break>;
    case <literal or type>:
        <statement>;
        <break>;
    ,
    ,
    default:
       <statement>;
       <break>;
}
```

Function Structure
```
<private, public, protected, internal> [static] <ReturnType>
<Function_Name>([Parameters])
{
    //body of the function;
    return <ReturnType>;
}
```

Sub Procedure Structure
```
<private, public, protected, internal> void <method_name>([Parameters])
{
    //body of the procedure;
}
```

Class Structure
```
public class <Class_Name>
{
    //body of class
}

public
'method_prototypes
'data_attributes
private
'method_prototypes
'data_attributes
internal
'method_prototypes
static
'method_prototypes
'data_attributes
```


Edited By:PsychoCoder, Martyr2
Published: October 9, 2007