



C#

IN A NUTSHELL

A Desktop Quick Reference

O'REILLY®

*Peter Drayton,
Ben Albahari & Ted Neward*



C#

IN A NUTSHELL

A Desktop Quick Reference



C#

IN A NUTSHELL

A Desktop Quick Reference

*Peter Drayton,
Ben Albahari & Ted Neward*

O'REILLY®

Beijing • Cambridge • Farnham • Köln • Paris • Sebastopol • Taipei • Tokyo



B

Format Specifiers

Table B-1 lists the numeric format specifiers supported by the `Format` method on the predefined numeric types.

Table B-1. Numeric Format Specifiers

Specifier	String result	Datatype
<code>C[n]</code>	<code>\$XX,XX.XX</code> <code>(\$XX,XXX.XX)</code>	Currency
<code>D[n]</code>	<code>[-]XXXXXXX</code>	Decimal
<code>E[n]</code> or <code>e[n]</code>	<code>[-]X.XXXXXXE+xxx</code> <code>[-]X.XXXXXXe+xxx</code> <code>[-]X.XXXXXXE-xxx</code> <code>[-]X.XXXXXXe-xxx</code>	Exponent
<code>F[n]</code>	<code>[-]XXXXXXX.XX</code>	Fixed point
<code>G[n]</code>	General or scientific	General
<code>N[n]</code>	<code>[-]XX,XXX.XX</code>	Number
<code>X[n]</code> or <code>x[n]</code>	Hex representation	Hex

This example uses numeric format specifiers without precision specifiers:

```
using System;
class TestDefaultFormats {
    static void Main() {
        int i = 654321;
        Console.WriteLine("{0:C}", i); // $654,321.00
        Console.WriteLine("{0:D}", i); // 654321
        Console.WriteLine("{0:E}", i); // 6.543210E+005
        Console.WriteLine("{0:F}", i); // 654321.00
        Console.WriteLine("{0:G}", i); // 654321
        Console.WriteLine("{0:N}", i); // 654,321.00
    }
}
```

```

        Console.WriteLine("{0:X}", i); // 9FBF1
        Console.WriteLine("{0:x}", i); // 9fbf1
    }
}

```

This example uses numeric format specifiers with precision specifiers on a variety of int values:

```

using System;
class TestIntegerFormats {
    static void Main() {
        int i = 123;
        Console.WriteLine("{0:C6}", i); // $123.000000
        Console.WriteLine("{0:D6}", i); // 000123
        Console.WriteLine("{0:E6}", i); // 1.230000E+002
        Console.WriteLine("{0:G6}", i); // 123
        Console.WriteLine("{0:N6}", i); // 123.000000
        Console.WriteLine("{0:X6}", i); // 00007B
        i = -123;
        Console.WriteLine("{0:C6}", i); // ($123.000000)
        Console.WriteLine("{0:D6}", i); // -000123
        Console.WriteLine("{0:E6}", i); // -1.230000E+002
        Console.WriteLine("{0:G6}", i); // -123
        Console.WriteLine("{0:N6}", i); // -123.000000
        Console.WriteLine("{0:X6}", i); // FFFF85
        i = 0;
        Console.WriteLine("{0:C6}", i); // $0.000000
        Console.WriteLine("{0:D6}", i); // 000000
        Console.WriteLine("{0:E6}", i); // 0.000000E+000
        Console.WriteLine("{0:G6}", i); // 0
        Console.WriteLine("{0:N6}", i); // 0.000000
        Console.WriteLine("{0:X6}", i); // 000000
    }
}

```

This example uses numeric format specifiers with precision specifiers on a variety of double values:

```

using System;
class TestDoubleFormats {
    static void Main() {
        double d = 1.23;
        Console.WriteLine("{0:C6}", d); // $1.230000
        Console.WriteLine("{0:E6}", d); // 1.230000E+000
        Console.WriteLine("{0:G6}", d); // 1.23
        Console.WriteLine("{0:N6}", d); // 1.230000
        d = -1.23;
        Console.WriteLine("{0:C6}", d); // ($1.230000)
        Console.WriteLine("{0:E6}", d); // -1.230000E+000
        Console.WriteLine("{0:G6}", d); // -1.23
        Console.WriteLine("{0:N6}", d); // -1.230000
        d = 0;
        Console.WriteLine("{0:C6}", d); // $0.000000
        Console.WriteLine("{0:E6}", d); // 0.000000E+000
        Console.WriteLine("{0:G6}", d); // 0
    }
}

```

```

        Console.WriteLine("{0:N6}", d); // 0.000000
    }
}

```

Picture Format Specifiers

Table B-2 lists the valid picture format specifiers supported by the `Format` method on the predefined numeric types (see the documentation for `System.IFormattable` in the .NET SDK).

Table B-2. Picture Format Specifiers

Specifier	String Result
0	Zero placeholder
#	Digit placeholder
.	Decimal point
,	Group separator or multiplier
%	Percent notation
E+, E-0 e+0, e-0	Exponent notation
\	Literal character quote
'xx' "xx"	Literal string quote
;	Section separator

This example uses picture-format specifiers on some `int` values:

```

using System;
class TestIntegerCustomFormats {
    static void Main() {
        int i = 123;
        Console.WriteLine("{0:#0}", i);           // 123
        Console.WriteLine("{0:#0;(#0)}", i);      // 123
        Console.WriteLine("{0:#0;(#0);<zero>}", i); // 123
        Console.WriteLine("{0:#%}", i);          // 12300%
        i = -123;
        Console.WriteLine("{0:#0}", i);           // -123
        Console.WriteLine("{0:#0;(#0)}", i);      // (123)
        Console.WriteLine("{0:#0;(#0);<zero>}", i); // (123)
        Console.WriteLine("{0:#%}", i);          // -12300%
        i = 0;
        Console.WriteLine("{0:#0}", i);           // 0
        Console.WriteLine("{0:#0;(#0)}", i);      // 0
        Console.WriteLine("{0:#0;(#0);<zero>}", i); // <zero>
        Console.WriteLine("{0:#%}", i);          // %
    }
}

```

The following example uses these picture format specifiers on a variety of double values:

```

using System;
class TestDoubleCustomFormats {

```

```

static void Main() {
    double d = 1.23;
    Console.WriteLine("{0:#.000E+00}", d); // 1.230E+00
    Console.WriteLine(
        "{0:#.000E+00;(#.000E+00})", d); // 1.230E+00
    Console.WriteLine(
        "{0:#.000E+00;(#.000E+00);<zero>}", d); // 1.230E+00
    Console.WriteLine("{0:##}", d); // 123%
    d = -1.23;
    Console.WriteLine("{0:#.000E+00}", d); // -1.230E+00
    Console.WriteLine(
        "{0:#.000E+00;(#.000E+00})", d); // (1.230E+00)
    Console.WriteLine(
        "{0:#.000E+00;(#.000E+00);<zero>}", d); // (1.230E+00)
    Console.WriteLine("{0:##}", d); // -123%
    d = 0;
    Console.WriteLine("{0:#.000E+00}", d); // 0.000E-01
    Console.WriteLine(
        "{0:#.000E+00;(#.000E+00})", d); // 0.000E-01
    Console.WriteLine(
        "{0:#.000E+00;(#.000E+00);<zero>}", d); // <zero>
    Console.WriteLine("{0:##}", d); // %
}
}

```

DateTime Format Specifiers

Table B-3 lists the valid format specifiers supported by the `Format` method on the `DateTime` type (see `System.IFormattable`).

Table B-3. *DateTime Format Specifiers*

Specifier	String Result
D	MM/dd/yyyy
d	dddd, MMMM dd, yyyy
f	dddd, MMMM dd, yyyy HH:mm
F	dddd, MMMM dd, yyyy HH:mm:ss
g	MM/dd/yyyy HH:mm
G	MM/dd/yyyy HH:mm:ss
m, M	MMMM dd
r, R	Ddd, dd MMM yyyy HH':'mm': 'ss 'GMT'
s	yyyy-MM-dd HH:mm:ss
S	yyyy-MM-dd HH:mm:ss GMT
t	HH:mm
T	HH:mm:ss
u	yyyy-MM-dd HH:mm:ss
U	dddd, MMMM dd, yyyy HH:mm:ss
y, Y	MMMM, yyyy

Here's an example that uses these custom format specifiers on a `DateTime` value:

```
using System;
class TestDateTimeFormats {
    static void Main() {
        DateTime dt = new DateTime(2000, 10, 11, 15, 32, 14);
        // Prints "2000-10-11T15:32:14"
        Console.WriteLine(dt.ToString());
        // Prints "Wednesday, October 11, 2000"
        Console.WriteLine("{0}", dt);
        // Prints "10/11/2000"
        Console.WriteLine("{0:d}", dt);
        // Prints "Wednesday, October 11, 2000"
        Console.WriteLine("{0:D}", dt);
        // Prints "Wednesday, October 11, 2000 3:32 PM"
        Console.WriteLine("{0:f}", dt);
        // Prints "Wednesday, October 11, 2000 3:32:14 PM"
        Console.WriteLine("{0:F}", dt);
        // Prints "10/11/2000 3:32 PM"
        Console.WriteLine("{0:g}", dt);
        // Prints "10/11/2000 3:32:14 PM"
        Console.WriteLine("{0:G}", dt);
        // Prints "October 11"
        Console.WriteLine("{0:m}", dt);
        // Prints "October 11"
        Console.WriteLine("{0:M}", dt);
        // Prints "Wed, 11 Oct 2000 22:32:14 GMT"
        Console.WriteLine("{0:r}", dt);
        // Prints "Wed, 11 Oct 2000 22:32:14 GMT"
        Console.WriteLine("{0:R}", dt);
        // Prints "3:32 PM"
        Console.WriteLine("{0:t}", dt);
        // Prints "3:32:14 PM"
        Console.WriteLine("{0:T}", dt);
        // Prints "2000-10-11 22:32:14Z"
        Console.WriteLine("{0:u}", dt);
        // Prints "Wednesday, October 11, 2000 10:32:14 PM"
        Console.WriteLine("{0:U}", dt);
        // Prints "October, 2000"
        Console.WriteLine("{0:y}", dt);
        // Prints "October, 2000"
        Console.WriteLine("{0:Y}", dt);
        // Prints "Wednesday the 11 day of October in the year 2000"
        Console.WriteLine(
            "{0:dddd 'the' d 'day of' MMMM 'in the year' yyyy}", dt);
    }
}
```