

This is Google's cache of http://tutorials.csharp-online.net/CSharp_Format_Specifiers%E2%80%94Numeric_Format_Specifiers. It is a snapshot of the page as it appeared on 20 May 2014 00:53:41 GMT. The [current page](#) could have changed in the meantime. [Learn more](#)

Tip: To quickly find your search term on this page, press **Ctrl+F** or **⌘-F** (Mac) and use the find bar.

[Text-only version](#)

C# Format Specifiers—Numeric Format Specifiers

Jump to: [navigation](#), [search](#)

Numeric 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
C[n]	\$XX,XX.XX (\$XX,XXX.XX)	Currency
D[n]	[-]XXXXXXXX	Decimal
E[n] or e[n]	[-]X.XXXXXXXE+xxx [-]X.XXXXXXXe+xxx [-]X.XXXXXXXE-xxx [-]X.XXXXXXXe-xxx	Exponent
F[n]	[-]XXXXXXXX.XX	Fixed point
G[n]	General or scientific	General
N[n]	[-]XX,XXX.XX	Number
X[n] or x[n]	Hex representation	Hex

[Visual C# Tutorials](#)

[Visual C# .NET Tutorials](#)

[C# Format Specifiers](#)

- **Numeric Format Specifiers**
- [Picture Format Specifiers](#)
- [DateTime Format Specifiers](#)

© 2006 O'Reilly & Assoc., Inc.

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
    }
}
```