

第7章 書式設定



書式設定は、決められた書式でデータを文字列として出力します。

書式設定の方法はいろいろあります。

書式指定文字列のなかの`{}`を利用する方法はすでに出てきているので、これについて解説します。

```
Console.WriteLine("{0,10}", 引数,...);
```

上の構文において、`{}`を書式指定項目といいます。これは、詳しく見ると

```
{index[,alignment][:formatString]}
```

`index`は当然省略不可です。パラメータ指定子といいます。

`alignment`は、省略可です。書式設定フィールドの幅を指定する符号付き整数値です。プラスは左揃え、マイナスは右揃えになります。

`formatString`は、省略可能で書式指定文字列といいます。これには、`Xnn`で表され、`X`を書式指定子(アルファベット)、`nn`は精度指定子(数字)といいます。`nn`は省略可能です。

標準書式指定文字	意味
Cまたはc	通過を表す文字列に変換。精度指定子は、小数点以下の桁数
Dまたはd	整数型のみ。整数の左側に精度指定子で指定した桁に達するまで0で埋められる
Eまたはe	1.2555E+128(または1.2555e+128)形式に変換。精度指定子は小数点以下の桁数。指数部は常に符号付き3桁。
Fまたはf	固定小数点数を表す文字列に変換。精度指定子は小数点以下の桁数。
Gまたはg	固定小数点または指数表記のうち簡潔な方を表示。精度指定子は、変換後の文字列の有効桁数。
Nまたはn	整数部は3桁ごとに区切り記号がはいる。精度指定子は小数部の桁数。
Pまたはp	パーセント値に変換。精度指定子は小数部の桁数。
Rまたはr	浮動小数点型のみ。返還後の文字列が変換前の数値に戻るよう解析される。精度指定子は無視される。
Xまたはx	整数型のみ。16進表記の文字列に変換。精度指定子は変換後の最小桁数。桁数に達しないときは0を左側に追加。

標準書式指定指定子で、希望の書式が得られないときはカスタム書式指定文字列を使います。

書式指定文字	名称	意味
0	Zero placeholder	0に対応する位置に数字がある時は、この数字が文字列にコピーされます。ない場合は0のままです。
#	Digit placeholder	#の位置に対応する数字がある時は、その数字が文字列にコピーされ、ない場合は何もコピーされない。
.	Decimal point	小数点の位置を指定。
,	桁区切り	(1)0または#の間に(,)があれば整数部が3桁ごとに(,)がはいる (2)0または#に挟まれない文字列内の整数部に(,)が1つ以上あれば、そのたびに1000で除される
%	Percentage placeholder	パーセント表示
E0,E+0,E-0	Scientific notation	小文字(e)も可。指数表記。0の数が指数部の最小の桁数。+があれば指数部に+または-がつく。-の時は指数部が負の時のみーを表示。+も-もなければ指数部が負の時のみ-を表示。
¥	Escape character	¥の後ろの文字がエスケープシーケンスとして解釈される。
'abc'または"abc"	リテラルとして出力文字列にコピーされる。	
その他		上記以外の文字は対応する位置にそのままコピーされる。

言葉で説明すると、かなり面倒なように思われますが、実際に使ってみるとたいしたことないものばかりです。

```
// format01.cs
```

```
using System;
```

```
class format01
```

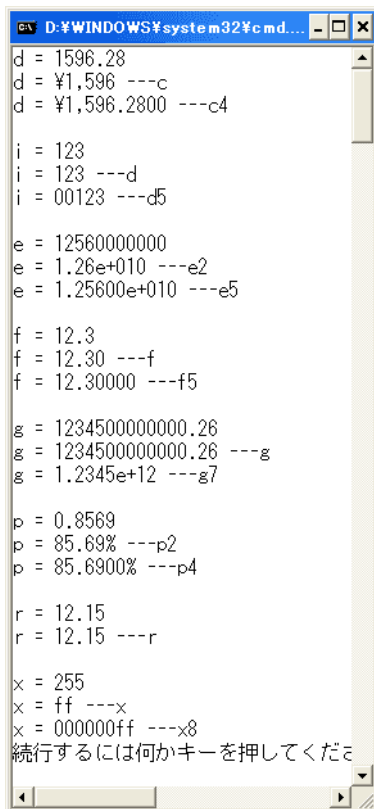
```
{  
    public static void Main()  
    {  
        double d = 1596.28;  
        int i = 123;  
        double e = 1.256E10;  
        double f = 12.3;  
        double g = 1234500000000.258;  
        double p = 0.8569;  
        double r = 12.15;  
        int x = 255;  
  
        Console.WriteLine("d = {0}", d);  
        Console.WriteLine("d = {0:c} ---c", d);  
        Console.WriteLine("d = {0:c4} ---c4", d);  
    }  
}
```

```

        Console.WriteLine();
        Console.WriteLine("i = {0}", i);
        Console.WriteLine("i = {0:d} ---d", i);
        Console.WriteLine("i = {0:d5} ---d5", i);
        Console.WriteLine();
        Console.WriteLine("e = {0}", e);
        Console.WriteLine("e = {0:e2} ---e2", e);
        Console.WriteLine("e = {0:e5} ---e5", e);
        Console.WriteLine();
        Console.WriteLine("f = {0}", f);
        Console.WriteLine("f = {0:f} ---f", f);
        Console.WriteLine("f = {0:f5} ---f5", f);
        Console.WriteLine();
        Console.WriteLine("g = {0}", g);
        Console.WriteLine("g = {0:g} ---g", g);
        Console.WriteLine("g = {0:g7} ---g7", g);
        Console.WriteLine();
        Console.WriteLine("p = {0}", p);
        Console.WriteLine("p = {0:p2} ---p2", p);
        Console.WriteLine("p = {0:p4} ---p4", p);
        Console.WriteLine();
        Console.WriteLine("r = {0}", r);
        Console.WriteLine("r = {0:r} ---r", r);
        Console.WriteLine();
        Console.WriteLine("x = {0}", x);
        Console.WriteLine("x = {0:x} ---x", x);
        Console.WriteLine("x = {0:x8} ---x8", x);
    }
}

```

実行結果は次の図のようになります。



```

D:\WINDOWS\system32\cmd...
d = 1596.28
d = ¥1,596 ---c
d = ¥1,596.2800 ---c4

i = 123
i = 123 ---d
i = 00123 ---d5

e = 12560000000
e = 1.26e+010 ---e2
e = 1.25600e+010 ---e5

f = 12.3
f = 12.30 ---f
f = 12.30000 ---f5

g = 1234500000000.26
g = 1234500000000.26 ---g
g = 1.2345e+12 ---g7

p = 0.8569
p = 85.69% ---p2
p = 85.6900% ---p4

r = 12.15
r = 12.15 ---r

x = 255
x = ff ---x
x = 000000ff ---x8

```

書式指定子cの場合は、整数部がコンマで区切られることに注意してください。

```
// format02.cs
```

```
using System;
```

```
class format02
```

```

{
    public static void Main()
    {
        int i = 123;
        double d = 123.45;
        double e = 10000000000;
    }
}

```

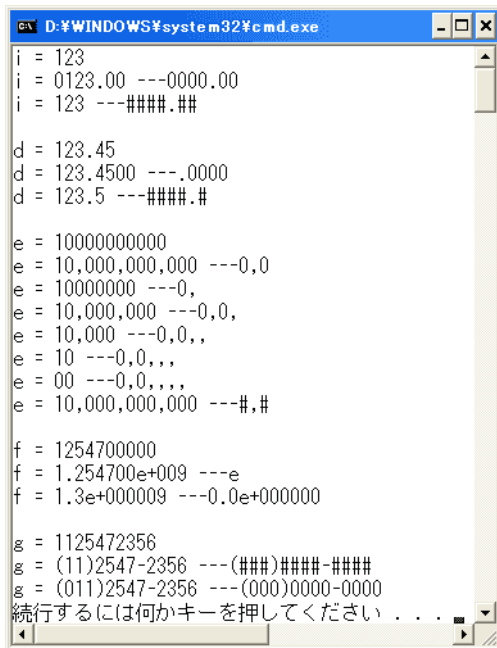
```

double f = 1254700000;
int g = 01125472356;

Console.WriteLine("i = {0}", i);
Console.WriteLine("i = {0:0000.00} ---0000.00", i);
Console.WriteLine("i = {0:####.##} ---####.##", i);
Console.WriteLine();
Console.WriteLine("d = {0}", d);
Console.WriteLine("d = {0:.0000} ---.0000", d);
Console.WriteLine("d = {0:####.#} ---####.#", d);
Console.WriteLine();
Console.WriteLine("e = {0}", e);
Console.WriteLine("e = {0:0,0} ---0,0", e);
Console.WriteLine("e = {0:0,} ---0,", e);
Console.WriteLine("e = {0:0,0,} ---0,0,", e);
Console.WriteLine("e = {0:0,0,,} ---0,0,,", e);
Console.WriteLine("e = {0:0,0,,,} ---0,0,,,", e);
Console.WriteLine("e = {0:0,0,,,,} ---0,0,,,,", e);
Console.WriteLine("e = {0:#,#} ---#,#", e);
Console.WriteLine();
Console.WriteLine("f = {0}", f);
Console.WriteLine("f = {0:e} ---e", f);
Console.WriteLine("f = {0:0.0e+000000} ---0.0e+000000", f);
Console.WriteLine();
Console.WriteLine("g = {0}", g);
Console.WriteLine("g = {0:(###)####-####} ---(###)####-####", g);
Console.WriteLine("g = {0:(000)0000-0000} ---(000)0000-0000", g);
}
}

```

実行結果を見てみましょう。



```

D:\WINDOWS\system32\cmd.exe
i = 123
i = 0123.00 ---0000.00
i = 123 ---####.##

d = 123.45
d = 123.4500 ---.0000
d = 123.5 ---####.#

e = 10000000000
e = 10,000,000,000 ---0,0
e = 10000000 ---0,
e = 10,000,000 ---0,0,
e = 10,000 ---0,0,,
e = 10 ---0,0,,,
e = 00 ---0,0,,,
e = 10,000,000,000 ---#,#

f = 1254700000
f = 1.254700e+009 ---e
f = 1.3e+000009 ---0.0e+000000

g = 1125472356
g = (11)2547-2356 ---(###)####-####
g = (011)2547-2356 ---(000)0000-0000
続行するには何かキーを押してください . . .

```

(.)の使い方がわかりにくいと思います。(実際のプログラミングではほとんど登場しないか...)