**Align String with Spaces [C#]**

This example shows how to align strings with spaces. The example formats text to table and writes it to console output.

To align string to the right or to the left use static method [String.Format](http://msdn2.microsoft.com/en-us/library/system.string.format.aspx). To **align string to the left** (spaces on the right) use formatting patern with **comma** (**,**) followed by a negative number of characters: String.Format(„{0,–10}“, text). To **right alignment** use a positive number: {0,10}.

Following example shows how to format text to the table. Values in the first and second column are aligned to the left and the third column is aligned to the right.

[C#]

Console.WriteLine("-------------------------------");

Console.WriteLine("First Name | Last Name | Age");

Console.WriteLine("-------------------------------");

Console.WriteLine(String.Format("{0,-10} | {1,-10} | {2,5}", "Bill", "Gates", 51));

Console.WriteLine(String.Format("{0,-10} | {1,-10} | {2,5}", "Edna", "Parker", 114));

Console.WriteLine(String.Format("{0,-10} | {1,-10} | {2,5}", "Johnny", "Depp", 44));

Console.WriteLine("-------------------------------");

Output string:

-------------------------------

First Name | Last Name | Age

-------------------------------

Bill | Gates | 51

Edna | Parker | 114

Johnny | Depp | 44

-------------------------------

**Indent String with Spaces [C#]**

This example shows how to indent strings using method for padding in C#. To **repeat spaces** use method [String.PadLeft](http://msdn2.microsoft.com/en-us/library/system.string.padleft.aspx). If you call „hello“.PadLeft(10) you will get the string aligned to the right: „     hello“. If you use empty string instead of the „hello“ string the result will be 10× repeated space character. This can be used to create simple Indent method.

The Indent method:

[C#]

public static string **Indent**(int count)

{

return "".PadLeft(count);

}

Test code:

[C#]

Console.WriteLine(Indent(0) + "List");

Console.WriteLine(Indent(3) + "Item 1");

Console.WriteLine(Indent(6) + "Item 1.1");

Console.WriteLine(Indent(6) + "Item 1.2");

Console.WriteLine(Indent(3) + "Item 2");

Console.WriteLine(Indent(6) + "Item 2.1");

Output string:

List

Item 1

Item 1.1

Item 1.2

Item 2

Item 2.1

# String Format for Int [C#]

Integer numbers can be formatted in .NET in many ways. You can use static method String.[Format](http://msdn2.microsoft.com/en-us/library/system.string.format.aspx) or instance method int.[ToString](http://msdn2.microsoft.com/en-us/library/system.int32.tostring.aspx). Following examples shows how to align numbers (with spaces or zeroes), how to format negative numbers or how to do custom formatting like phone numbers.

## Add zeroes before number

To add zeroes before a number, use **colon separator „:“** and write as many zeroes as you want.

[C#]

String.Format("{0:00000}", 15); // "00015"

String.Format("{0:00000}", -15); // "-00015"

## Align number to the right or left

To align number to the right, use **comma „,“** followed by a number of characters. This alignment option must be before the colon separator.

[C#]

String.Format("{0,5}", 15); // " 15"

String.Format("{0,-5}", 15); // "15 "

String.Format("{0,5:000}", 15); // " 015"

String.Format("{0,-5:000}", 15); // "015 "

## Different formatting for negative numbers and zero

You can have special format for negative numbers and zero. Use **semicolon separator „;“** to separate formatting to two or three sections. The second section is format for negative numbers, the third section is for zero.

[C#]

String.Format("{0:#;minus #}", 15); // "15"

String.Format("{0:#;minus #}", -15); // "minus 15"

String.Format("{0:#;minus #;zero}", 0); // "zero"

## Custom number formatting (e.g. phone number)

Numbers can be formatted also to any custom format, e.g. like phone numbers or serial numbers.

[C#]

String.Format("{0:+### ### ### ###}", 447900123456); // "+447 900 123 456"

String.Format("{0:##-####-####}", 8958712551); // "89-5871-2551"

# String Format for Double [C#]

The following examples show how to format float numbers to string in C#. You can use static method [**String.Format**](http://msdn2.microsoft.com/en-us/library/system.string.format.aspx) or instance methods [double.ToString](http://msdn2.microsoft.com/en-us/library/kfsatb94.aspx) and [float.ToString](http://msdn2.microsoft.com/en-us/library/f71z6k0c.aspx).

## Digits after decimal point

This example formats double to string with **fixed number of decimal places**. For two decimal places use pattern „**0.00**“. If a float number has less decimal places, the rest digits on the right will be zeroes. If it has more decimal places, the number will be rounded.

[C#]

// just two decimal places

String.Format("{0:0.00}", 123.4567); // "123.46"

String.Format("{0:0.00}", 123.4); // "123.40"

String.Format("{0:0.00}", 123.0); // "123.00"

Next example formats double to string with **floating number of decimal places**. E.g. for maximal two decimal places use pattern „**0.##**“.

[C#]

// max. two decimal places

String.Format("{0:0.##}", 123.4567); // "123.46"

String.Format("{0:0.##}", 123.4); // "123.4"

String.Format("{0:0.##}", 123.0); // "123"

## Digits before decimal point

If you want a float number to have any **minimal number of digits before decimal point** use N-times zero before decimal point. E.g. pattern „**00.0**“ formats a float number to string with at least two digits before decimal point and one digit after that.

[C#]

// at least two digits before decimal point

String.Format("{0:00.0}", 123.4567); // "123.5"

String.Format("{0:00.0}", 23.4567); // "23.5"

String.Format("{0:00.0}", 3.4567); // "03.5"

String.Format("{0:00.0}", -3.4567); // "-03.5"

## Thousands separator

To format double to string **with use of thousands separator** use zero and comma separator before an usual float formatting pattern, e.g. pattern „**0,0.0**“ formats the number to use thousands separators and to have one decimal place.

[C#]

String.Format("{0:0,0.0}", 12345.67); // "12,345.7"

String.Format("{0:0,0}", 12345.67); // "12,346"

## Zero

Float **numbers between zero and one** can be formatted in two ways, with or without leading zero before decimal point. To format number without a leading zero use **#** before point. For example „**#.0**“ formats number to have one decimal place and zero to N digits before decimal point (e.g. „.5“ or „123.5“).

Following code shows **how can be formatted a zero** (of double type).

[C#]

String.Format("{0:0.0}", 0.0); // "0.0"

String.Format("{0:0.#}", 0.0); // "0"

String.Format("{0:#.0}", 0.0); // ".0"

String.Format("{0:#.#}", 0.0); // ""

## Align numbers with spaces

To align float number **to the right** use comma „**,**“ option before the colon. Type comma followed by a number of spaces, e.g. „**0,10:0.0**“ (this can be used only in [String.Format](http://msdn2.microsoft.com/en-us/library/system.string.format.aspx) method, not in [double.ToString](http://msdn2.microsoft.com/en-us/library/kfsatb94.aspx) method). To align numbers **to the left** use negative number of spaces.

[C#]

String.Format("{0,10:0.0}", 123.4567); // " 123.5"

String.Format("{0,-10:0.0}", 123.4567); // "123.5 "

String.Format("{0,10:0.0}", -123.4567); // " -123.5"

String.Format("{0,-10:0.0}", -123.4567); // "-123.5 "

## Custom formatting for negative numbers and zero

If you need to use custom format for negative float numbers or zero, use **semicolon separator** „**;**“ to split pattern to **three sections**. The first section formats positive numbers, the **second section formats negative numbers** and the third section formats zero. If you omit the last section, zero will be formatted using the first section.

[C#]

String.Format("{0:0.00;minus 0.00;zero}", 123.4567); // "123.46"

String.Format("{0:0.00;minus 0.00;zero}", -123.4567); // "minus 123.46"

String.Format("{0:0.00;minus 0.00;zero}", 0.0); // "zero"

## Some funny examples

As you could notice in the previous example, you can put any text into formatting pattern, e.g. before an usual pattern „***my text* 0.0**“. You can even put any text between the zeroes, e.g. „**0*aaa*.*bbb*0**“.

[C#]

String.Format("{0:my number is 0.0}", 12.3); // "my number is 12.3"

String.Format("{0:0aaa.bbb0}", 12.3); // "12aaa.bbb3"

# String Format for DateTime [C#]

This example shows how to format [DateTime](http://msdn2.microsoft.com/en-us/library/system.datetime.aspx) using [String.Format](http://msdn2.microsoft.com/en-us/library/system.string.format.aspx) method. All formatting can be done also using [DateTime.ToString](http://msdn2.microsoft.com/en-us/library/zdtaw1bw.aspx) method.

## Custom DateTime Formatting

There are following custom format specifiers y (year), M (month), d (day), h (hour 12), H (hour 24), m (minute), s (second), f (second fraction), F (second fraction, trailing zeroes are trimmed), t (P.M or A.M) and z (time zone).

Following examples demonstrate how are the format specifiers rewritten to the output.

[C#]

// create date time 2008-03-09 16:05:07.123

DateTime dt = new DateTime(2008, 3, 9, 16, 5, 7, 123);

String.Format("{0:y yy yyy yyyy}", dt); // "8 08 008 2008" year

String.Format("{0:M MM MMM MMMM}", dt); // "3 03 Mar March" month

String.Format("{0:d dd ddd dddd}", dt); // "9 09 Sun Sunday" day

String.Format("{0:h hh H HH}", dt); // "4 04 16 16" hour 12/24

String.Format("{0:m mm}", dt); // "5 05" minute

String.Format("{0:s ss}", dt); // "7 07" second

String.Format("{0:f ff fff ffff}", dt); // "1 12 123 1230" sec.fraction

String.Format("{0:F FF FFF FFFF}", dt); // "1 12 123 123" without zeroes

String.Format("{0:t tt}", dt); // "P PM" A.M. or P.M.

String.Format("{0:z zz zzz}", dt); // "-6 -06 -06:00" time zone

You can use also **date separator** / (slash) and **time sepatator** : (colon). These characters will be rewritten to characters defined in the current [DateTimeForma­tInfo.DateSepa­rator](http://msdn2.microsoft.com/en-us/library/system.globalization.datetimeformatinfo.dateseparator.aspx) and [DateTimeForma­tInfo.TimeSepa­rator](http://msdn2.microsoft.com/en-us/library/system.globalization.datetimeformatinfo.timeseparator.aspx).

[C#]

// date separator in german culture is "." (so "/" changes to ".")

String.Format("{0:d/M/yyyy HH:mm:ss}", dt); // "9/3/2008 16:05:07" - english (en-US)

String.Format("{0:d/M/yyyy HH:mm:ss}", dt); // "9.3.2008 16:05:07" - german (de-DE)

Here are some examples of custom date and time formatting:

[C#]

// month/day numbers without/with leading zeroes

String.Format("{0:M/d/yyyy}", dt); // "3/9/2008"

String.Format("{0:MM/dd/yyyy}", dt); // "03/09/2008"

// day/month names

String.Format("{0:ddd, MMM d, yyyy}", dt); // "Sun, Mar 9, 2008"

String.Format("{0:dddd, MMMM d, yyyy}", dt); // "Sunday, March 9, 2008"

// two/four digit year

String.Format("{0:MM/dd/yy}", dt); // "03/09/08"

String.Format("{0:MM/dd/yyyy}", dt); // "03/09/2008"

## Standard DateTime Formatting

In [DateTimeForma­tInfo](http://msdn2.microsoft.com/en-us/library/system.globalization.datetimeformatinfo.aspx) there are defined standard patterns for the current culture. For example property [ShortTimePattern](http://msdn2.microsoft.com/en-us/library/system.globalization.datetimeformatinfo.shorttimepattern.aspx) is string that contains value h:mm tt for **en-US** culture and value HH:mm for **de-DE** culture.

Following table shows patterns defined in [DateTimeForma­tInfo](http://msdn2.microsoft.com/en-us/library/system.globalization.datetimeformatinfo.aspx) and their values for en-US culture. First column contains format specifiers for the [String.Format](http://msdn2.microsoft.com/en-us/library/system.string.format.aspx) method.

|  |  |  |
| --- | --- | --- |
| **Specifier** | **DateTimeFormatInfo property** | **Pattern value (for en-US culture)** |
| t | ShortTimePattern | h:mm tt |
| d | ShortDatePattern | M/d/yyyy |
| T | LongTimePattern | h:mm:ss tt |
| D | LongDatePattern | dddd, MMMM dd, yyyy |
| f | (combination of *D* and *t*) | dddd, MMMM dd, yyyy h:mm tt |
| F | FullDateTimePattern | dddd, MMMM dd, yyyy h:mm:ss tt |
| g | (combination of *d* and *t*) | M/d/yyyy h:mm tt |
| G | (combination of *d* and *T*) | M/d/yyyy h:mm:ss tt |
| m, M | MonthDayPattern | MMMM dd |
| y, Y | YearMonthPattern | MMMM, yyyy |
| r, R | RFC1123Pattern | ddd, dd MMM yyyy HH':'mm':'ss 'GMT' (\*) |
| s | SortableDateTi­mePattern | yyyy'-'MM'-'dd'T'HH':'mm':'ss (\*) |
| u | UniversalSorta­bleDateTimePat­tern | yyyy'-'MM'-'dd HH':'mm':'ss'Z' (\*) |
|  |  | (\*) = culture independent |

Following examples show usage of **standard format specifiers** in [String.Format](http://msdn2.microsoft.com/en-us/library/system.string.format.aspx) method and the resulting output.

[C#]

String.Format("{0:t}", dt); // "4:05 PM" ShortTime

String.Format("{0:d}", dt); // "3/9/2008" ShortDate

String.Format("{0:T}", dt); // "4:05:07 PM" LongTime

String.Format("{0:D}", dt); // "Sunday, March 09, 2008" LongDate

String.Format("{0:f}", dt); // "Sunday, March 09, 2008 4:05 PM" LongDate+ShortTime

String.Format("{0:F}", dt); // "Sunday, March 09, 2008 4:05:07 PM" FullDateTime

String.Format("{0:g}", dt); // "3/9/2008 4:05 PM" ShortDate+ShortTime

String.Format("{0:G}", dt); // "3/9/2008 4:05:07 PM" ShortDate+LongTime

String.Format("{0:m}", dt); // "March 09" MonthDay

String.Format("{0:y}", dt); // "March, 2008" YearMonth

String.Format("{0:r}", dt); // "Sun, 09 Mar 2008 16:05:07 GMT" RFC1123

String.Format("{0:s}", dt); // "2008-03-09T16:05:07" SortableDateTime

String.Format("{0:u}", dt); // "2008-03-09 16:05:07Z" UniversalSortableDateTime