

## Best way to repeat a character in C#

[Ask Question](#)

What it's the best way to generate a string of `\t` 's in C#

675

I am learning C# and experimenting with different ways of saying the same thing.



`Tabs(uint t)` is a function that returns a `string` with `t` amount of `\t` 's



94

For example `Tabs(3)` returns `"\t\t\t"`

Which of these three ways of implementing `Tabs(uint numTabs)` is best?

Of course that depends on what "best" means.

1. The LINQ version is only two lines, which is nice. But are the calls to `Repeat` and `Aggregate` unnecessarily time/resource consuming?
2. The `StringBuilder` version is very clear but is the `StringBuilder` class somehow slower?
3. The `string` version is basic, which means it is easy to understand.
4. Does it not matter at all? Are they all equal?

These are all questions to help me get a better feel for C#.

```
private string Tabs(uint numTabs)
{
    IEnumerable<string> tabs = Enumerable.Repeat("\t", (int) numTabs);
    return (numTabs > 0) ? tabs.Aggregate((sum, next) => sum + next) : "";
}

private string Tabs(uint numTabs)
{
    StringBuilder sb = new StringBuilder();
    for (uint i = 0; i < numTabs; i++)
        sb.Append("\t");

    return sb.ToString();
}

private string Tabs(uint numTabs)
{
    string output = "";
    for (uint i = 0; i < numTabs; i++)
    {
        output += '\t';
    }
    return output;
}
```

c#

.net

string

edited Jan 1 '14 at 2:49

**i3arnon****81.6k** 20 224 267

asked Jan 4 '09 at 21:56

**Alex Baranosky****21k** 35 88 142

Home

PUBLIC

Stack Overflow

Tags

Users

Jobs

19 Answers

What about this:

Teams



Teams

Q&amp;A for work

Learn More

1291

```
string tabs = new String('\t', n);
```



Where `n` is the number of times you want to repeat the string.



Or better:

```
static string Tabs(int n)
{
    return new String('\t', n);
}
```

edited Apr 13 '16 at 8:36



[silkfire](#)

13.7k 7 54 71

answered Jan 4 '09 at 22:00



[CMS](#)

600k 162 847 815

is there any way to repeat this for a word? instead of using '\t' how to use "time" – [asdfkjaskdfjk](#) Apr 3 '13 at 12:41

See further down: [stackoverflow.com/a/3097925/281077](https://stackoverflow.com/a/3097925/281077) – [Paaland](#) Apr 4 '13 at 8:18

6 [@user1478137](#) Or, better (also further down): [this](#) – [Xynariz](#) Apr 8 '14 at 19:51

117

```
string.Concat(Enumerable.Repeat("ab", 2));
```

Returns

"abab"

And

```
string.Concat(Enumerable.Repeat("a", 2));
```

Returns

"aa"

from...

[Is there a built-in function to repeat string or char in .net?](#)

edited May 23 '17 at 12:26



Community ♦

1 1

answered Mar 13 '13 at 16:16



Carter Medlin

8,681 4 46 61

Make it better by doing it without Linq and with `StruingBuilder` ! –

[Bitterblue](#) Apr 15 '14 at 15:29 ✎

4 "StringBuilder" is not necessarily faster

[stackoverflow.com/questions/585860/...](https://stackoverflow.com/questions/585860/...) – [Schiavini](#) Dec 9 '14 at 16:13 ✎

4 It may not be faster, but it can save on memory allocations (pressure) if you specify the correct capacity up front, and that can be as important as the micro benchmark of profiling this. – [wekempf](#) Apr 5 '16 at 13:03

This is really cool solution! – [sabiland](#) Feb 20 '18 at 7:34

```
var strRepeat = string.Concat(Enumerable.Repeat("ABC", 1000000));
//50ms – yongfa365 Dec 7 '18 at 2:31
```

▲ In all versions of .NET, you can repeat a string thus:

114 

```
public static string Repeat(string value, int count)
{
    return new StringBuilder(value.Length * count).Insert(0, value, 0, value.Length);
}
```

▼ To repeat a character, `new String('\t', count)` is your best bet. See [the answer by @CMS](#).

edited May 23 '17 at 12:03



Community ♦

1 1

answered Apr 6 '09 at 10:38



[Binoj Antony](#)

12.8k 24 82 94

6 This is by far the fastest performance wise for String. Other methods create too much overhead. – [midspace](#) May 11 '15 at 0:22

▲ The best version is certainly to use the builtin way:

60 

```
string Tabs(int len) { return new string('\t', len); }
```

▼ Of the other solutions, prefer the easiest; only if this is proving too slow, strive for a more efficient solution.

~~If you use a `StringBuilder` and know its resulting length in advance, then also use an appropriate constructor, this is much more efficient because it means that only one time-consuming allocation takes place, and no unnecessary copying of data.~~ Nonsense: of course the above code is more efficient.

edited Aug 23 '11 at 17:31

answered Jan 4 '09 at 22:01



**Konrad Rudolph**

**401k** 101 790 1038

12 `sb.Append('\t', len);` – [dan-gph](#) Jun 3 '10 at 2:27

7 My benchmarks are showing `new string('\t', len)` to be between 2x and 7x faster than the `StringBuilder` approach. Why do you think `StringBuilder` is more efficient in this case? – [StriplingWarrior](#) Aug 23 '11 at 16:32

6 @StriplingWarrior Thanks for correcting this (after it has been standing here for two years, no less!). – [Konrad Rudolph](#) Aug 23 '11 at 17:30 ✎

Extension methods:

48

```
public static string Repeat(this string s, int n)
{
    return new String(Enumerable.Range(0, n).SelectMany(x => s).ToArray());
}

public static string Repeat(this char c, int n)
{
    return new String(c, n);
}
```

edited Feb 7 '14 at 10:35



bluish

14.2k 16 94 149

answered Jun 22 '10 at 23:34



Rodrick Chapman

4,722 24 30

35 Once upon a time people used while loops – [prabhakaran](#) Sep 26 '13 at 11:31

2 nice answer, helped me more than `new String('\t', 2) ...` – [AceMark](#) Sep 27 '13 at 1:18

2 @prabhakaran I agree but the advantage of Linq is in communication; it's almost always trivial to reimplement a Linq expression using imperative constructs. – [Rodrick Chapman](#) Oct 7 '13 at 4:14

10 `Enumerable.Range(0, n).SelectMany(x => s)` can be replaced by a simple `Enumerable.Repeat(s, n)` . – [Palec](#) Apr 9 '16 at 15:10

1 @Palec No it cannot. The `SelectMany` will for each single dummy `x` give a sequence of `char` values (since the `string s` implements `IEnumerable<char>`), and all those `char` sequences are concatenated to one long `char` sequence. What you suggest will instead give an `IEnumerable<string>` . It is not the same. – [Jeppe Stig Nielsen](#) Dec 20 '17 at 22:37

What about using extension method?

40

```
public static class StringExtensions
{
    public static string Repeat(this char chatToRepeat, int repeat) {
        return new string(chatToRepeat,repeat);
    }
    public static string Repeat(this string stringToRepeat,int repea
```

```

{
    var builder = new StringBuilder(repeat*stringToRepeat.Length)
    for (int i = 0; i < repeat; i++) {
        builder.Append(stringToRepeat);
    }
    return builder.ToString();
}

```

You could then write :

```

Debug.WriteLine('-'.Repeat(100)); // For Chars
Debug.WriteLine("Hello".Repeat(100)); // For Strings

```

Note that a performance test of using the stringBuilder version for simple characters instead of strings gives you a major performance penalty : on my computer the difference in measured performance is 1:20 between: `Debug.WriteLine('-'.Repeat(1000000))` //char version and `Debug.WriteLine("-".Repeat(1000000))` //string version

edited Jun 20 '11 at 14:24



[larsmoa](#)

9,171 3 49 75

answered Jan 27 '09 at 12:28



[Ronnie](#)

2,664 9 39 66

- 
- 2 The performance of the string version may be improved by using [StringBuilder.Insert\(Int32, String, Int32\)](#), as [Binoj Anthony answered](#). – [Palec](#) Apr 9 '16 at 15:00
- 

How about this:



22

```
//Repeats a character specified number of times
public static string Repeat(char character,int numberOfIterations)
{
    return "".PadLeft(numberOfIterations, character);
}

//Call the Repeat method
Console.WriteLine(Repeat('\t',40));
```

edited Feb 7 '14 at 10:35



bluish

14.2k 16 94 149

answered Jun 20 '11 at 13:50



Denys Wessels

221 2 2

- 4 haha.. that's what I could think of when I fell into this situation once. But personally I found `new string('\t', 10)` to be the best solution – [shashwat](#) Nov 18 '13 at 11:27

This trick is also used in [Essential C# 6.0](#). – [The Inventor of God](#) Jan 25 '17 at 10:16



I know that this question is five years old already but there is a simple way to repeat a string that even works in .Net 2.0.

19

To repeat a string:

```
string repeated = new String('+', 3).Replace("+", "Hello, ");
```

Returns

```
"Hello, Hello, Hello, "
```

To repeat a string as an array:

```
// Two line version.
string repeated = new String('+', 3).Replace("+", "Hello,");
string[] repeatedArray = repeated.Split(',');

// One line version.
string[] repeatedArray = new String('+', 3).Replace("+", "Hello,").Split(',');
```

Returns

```
{"Hello", "Hello", "Hello", ""}
```

Keep it simple.

edited Aug 23 '15 at 22:37

answered Dec 3 '14 at 2:42



ChaimG

2,339 2 17 30



Let's say you want to repeat 't' n number of times, you can use;

16

```
String.Empty.PadRight(n, '\t')
```



answered Aug 28 '13 at 10:45




Amin

236 2 4

Thanks. This is also very fast.

[codereview.stackexchange.com/questions/36391/...](https://codereview.stackexchange.com/questions/36391/...) – Sunsetquest Apr 8 '16 at 22:03

I have always done it this way. Even if wrapped in an extension method it is simpler than other implementations posted here. Note that OP did just want to repeat \t, not strings. – [Michael Ribbons](#) Apr 29 '16 at 2:18 

▲ Your first example which uses `Enumerable.Repeat` :

15



```
private string Tabs(uint numTabs)
{
    IEnumerable<string> tabs = Enumerable.Repeat(
        "\t", (int) numTabs);
    return (numTabs > 0) ?
        tabs.Aggregate((sum, next) => sum + next) : "";
}
```

can be rewritten more compactly with `String.Concat` :

```
private string Tabs(uint numTabs)
{
    return String.Concat(Enumerable.Repeat("\t", (int) numTabs));
}
```

edited Jan 5 '16 at 14:22



[Wai Ha Lee](#)

6,056 12 40 65

answered Jan 26 '11 at 2:43



[Ray Vega](#)

82.1k 91 200 194



Using `String.Concat` and `Enumerable.Repeat` which will be less expensive than using `String.Join`

14



```
public static Repeat(this String pattern, int count)
{
    return String.Concat(Enumerable.Repeat(pattern, count));
}
```

answered Apr 23 '13 at 7:09

[bradgonesurfing](#)

16.4k 10 84 151



```
var str = new string(Enumerable.Repeat('\t', numTabs).ToArray());
```

8



answered Feb 7 '14 at 11:07

[w.b](#)

8,957 4 20 38



The answer really depends on the complexity you want. For example, I want to outline all my indents with a vertical bar, so my indent string is determined as follows:

4



```
return new string(Enumerable.Range(0, indentSize*indent).Select(
    n => n%4 == 0 ? '|' : ' ').ToArray());
```

answered Apr 6 '09 at 10:58

[Dmitri Nesteruk](#)

11.1k 17 79 139

▲ You can create an extension method

2

▼

```
static class MyExtensions
{
    internal static string Repeat(this char c, int n)
    {
        return new string(c, n);
    }
}
```

Then you can use it like this

```
Console.WriteLine('\t'.Repeat(10));
```

answered Feb 16 '18 at 17:56



ivan

48 3

▲ And yet another method

1

▼

```
new System.Text.StringBuilder().Append('\t', 100).ToString()
```

answered Jan 22 '18 at 8:31



Artyom

1,967 19 44

---

Passing the resulting string length to `StringBuilder` constructor saves reallocation. Still, this is more chatty and less efficient than using the `string` constructor that can repeat a given character, as already shown in the accepted answer. – [Palec](#) Jan 22 '18 at 13:48

---

@Palec you're right. Still it is another answer – [Artyom](#) Jan 22 '18 at 18:16

---



1



Without a doubt the accepted answer is the best and fastest way to repeat a single character.

Binoj Anthony's answer is a simple and quite efficient way to repeat a string.

However, if you don't mind a little more code, you can use my array fill technique to efficiently create these strings even faster. In my comparison tests, the code below executed in about 35% of the time of the `StringBuilder.Insert` code.

```
public static string Repeat(this string value, int count)
{
    var values = new char[count * value.Length];
    values.Fill(value.ToCharArray());
    return new string(values);
}

public static void Fill<T>(this T[] destinationArray, params T[] values)
{
    if (destinationArray == null)
    {
        throw new ArgumentNullException("destinationArray");
    }

    if (value.Length > destinationArray.Length)
    {
        throw new ArgumentException("Length of value array must not be greater than length of destination");
    }

    // set the initial array value
    Array.Copy(value, destinationArray, value.Length);

    int copyLength, nextCopyLength;

    for (copyLength = value.Length; (nextCopyLength = copyLength << 2) < destinationArray.Length; copyLength = nextCopyLength)
    {
        Array.Copy(destinationArray, 0, destinationArray, copyLength, copyLength);
    }
}
```

```
    Array.Copy(destinationArray, 0, destinationArray, copyLength,
destinationArray.Length - copyLength);
}
```

For more about this array fill technique, see [Fastest way to fill an array with a single value](#)

answered Aug 24 '18 at 21:12



Grax

2,821 12 21

For me is fine:

1

```
public static class Utils
{
    public static string LeftZerosFormatter(int zeros, int val)
    {
        string valstr = val.ToString();

        valstr = new string('0', zeros) + valstr;

        return valstr.Substring(valstr.Length - zeros, zeros);
    }
}
```

edited Sep 5 '18 at 11:57

answered Jun 4 '18 at 9:39



Ángel Ibáñez

77 5

Try this:

0

1. Add Microsoft.VisualBasic reference
2. Use: `String result = Microsoft.VisualBasic.Strings.StrDup(5,"hi");`
3. Let me know if it works for you.

edited Dec 22 '17 at 16:21



LarsTech

70.4k 12 108 160

answered Dec 22 '17 at 16:06



Toso Pankovski

1

Adding dependence on another assembly and the need to load it is unreasonable in most cases when you need such functionality. Still, good to know there is such a thing in .NET. – [Palec](#) Dec 24 '17 at 15:58

There many pros and cons in different contexts, so everyone should decide depending on the situation she/he is in. – [Toso Pankovski](#) Dec 25 '17 at 19:01

Albeit very similar to a previous suggestion, I like to keep it simple and apply the following:

-1

```
string MyFancyString = "*";
int strLength = 50;
System.Console.WriteLine(MyFancyString.PadRight(strLength, "*"));
```

Standard .Net really,

answered Aug 27 '18 at 7:28



Porky

157 1 4



It will just add padding, does not repeat – [levi](#) Feb 6 at 21:08

---

---