Strings in C#

Two basic types of strings

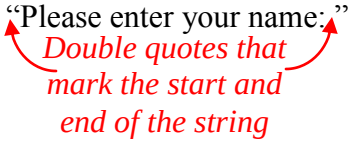
String literal

Anytime you can look at the code and **literally** read the text of the string it is a string literal.

The text of the string is "marked" by double quotes. **Anytime** the compiler sees a double quote (") it sets **everything** after it aside as a string **until** it hits the next double quote.

Examples:

- "Hi Mom!"
- "Please enter your name: "



String variable

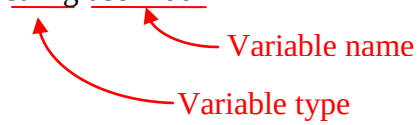
A "bucket" that can hold any string. You can put different strings in it. The strings that it can hold **vary**.

When you create a variable you have to tell the compiler two things:

1. The **type** of information the variable can hold.
2. The **name** that will be used to refer to and use the variable.

Examples:

- string myStr
- string userAddr



How to use strings

Display in the console window

To display a string in the console window, use the Console **class** provided by .NET. The Console class has two **methods** for displaying strings:

- Console.WriteLine(*string*) – display the *string* and "hit" the enter key moving the cursor down to the start of the next line.
- Console.Write(*string*) – display the *string* but just stop "typing" and leave the cursor at the end of the line ready to add more text to the line.

Strings in C#

Combining strings

There are two basic ways to combine strings:

1. "**Add**" them together using the plus (+) symbol. You can add string literals and variables in any combination.

   Example (given that userName & userAddr are string variables):

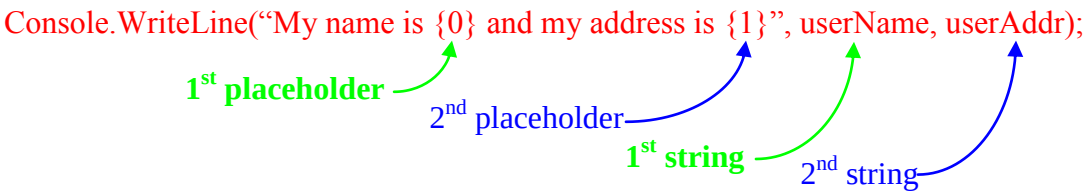
   "My name is " + userName + " and my address is " + userAddr

   If the string variable userName held the string "Mr. Thompson" and the string variable userAddr held the string "603 N. 179th St" then the above would produce the following single string:

   My name is Mr. Thompson and my address is 603 N. 179th St

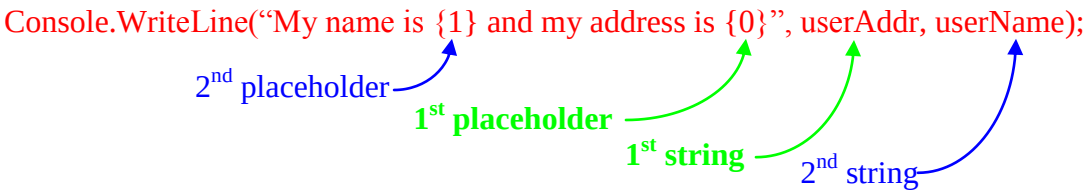
   In the first red line where the strings are being added together, note the space at the end of each string literal…they are there so that the text from the string variables are not displayed right after the last letter of the string literal.

2. **Console Write and WriteLine Placeholders**. The Write and WriteLine methods of the Console class have a special capability for combining strings. These two methods use curly brackets { } to mark "placeholders" they will use to "plug-in" other strings. You can think of this as a kind of string template that says "put a string here and put another string there." You can have as many placeholders as you like in a string. They are numbered, starting with zero. Here is an example:



   This will produce exactly the same output as the "add" example above.

   Now here is a weird twist: the numbers in the placeholder tell the compiler which of the following strings to use. You **don't** have to **use** zero first **but** you need to order the following replacement strings in the right order, the "zeroth" one first. Here is the above example rearranged…note how **both** the placeholder numbers and the replacement strings are reversed. This will produce **exactly** the same output as the above example!



Strings in C#

Putting "special" characters in strings

Question for you: if you mark the start and the end of a string literal with double quotes, what if you want to display a string with a double quote in it? How can you do that? Consider the following:

```
Console.WriteLine("Mr. Thompson said "Writing software is fun!"");
```

We would like that to display the following on the console window:

```
Mr. Thompson said "Writing software is fun!"
```

But…it won't. As soon as the compiler sees the 2nd double quote (which we mean to be the start of Mr. Thompson said) it will say oh! I see the end of this string literal. Oops. That isn't what we want.

There is a way of putting characters that have special meaning for strings into a string and have it be treated as a "normal" character. If you put a backslash right before the special character, it will treat is as a normal character! Here is what it would look like:

These special characters are called "**control characters**" and putting the backslash before them is called "**escaping**" the control character.

Most common control characters

Here is a list of the most common control characters and what they do:

Control character	Effect – what appears in the string
\t	Tab
\n	Newline – hitting 'enter'
\"	Double quote
\\	Backslash