**sprintf function**



Original tutorial at: <https://arduinobasics.blogspot.com/2019/05/sprintf-function.html>

**Description**

This tutorial will help you to understand the **sprintf** function, and how to use it. Essentially, the sprintf function allows you to construct a string using a pre-formatted string template to which you can insert variables at pre-defined locations. The sprintf function will "compile" the string and assign it to a char array. All you have to do is make sure that the char array is large enough to hold all of the characters in the string. The best way to understand the sprintf function is with examples. And luckily, I have examples. What are we waiting for ? Let's dive in.

**PLEASE NOTE:  
  
SOME FORMAT SPECIFIERS (eg. Floats, Scientific, and long long int)  
WILL NOT WORK ON ARDUINO.  
OTHER FORMAT SPECIFIERS SHOULD BE FINE.**

**Parts Required:** an Arduino or ESP32 and a USB cable.

**sprintf** ( char\* array,   const char\* strTemplate,   var1...);

**Arduino IDE**

While there are many [Arduino IDE alternatives](https://arduinobasics.blogspot.com/p/arduino-ide.html) out there, I would recommend that you use the [official Arduino IDE](https://www.arduino.cc/en/Main/Software) for this project. I used the official Arduino IDE app (v1.8.5) for Windows 10.  
Make sure to get the most up-to-date version for your operating system [here.](https://www.arduino.cc/en/Main/Software)

[](https://www.arduino.cc/en/Main/Software)

**Arduino Code**

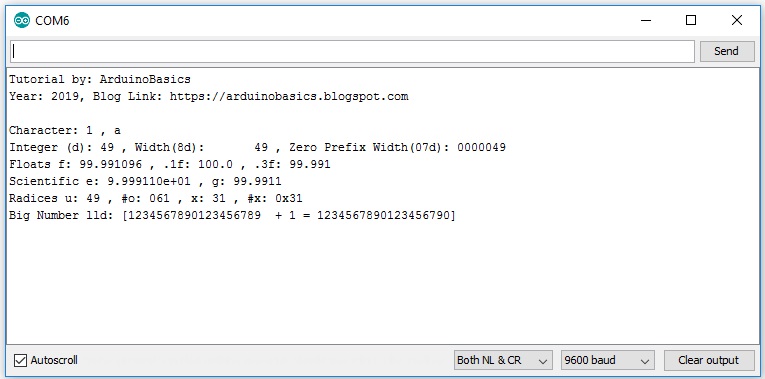
The code below will show you how to use the sprintf function and includes a number of different format specifiers to play with. In each case the sprintf function writes to the "data" character array, and subsequently sends it through to the Serial monitor. The string template helps to construct the data output, allowing you to insert variables at specific locations within the text. The format of the variable is defined by the "format specifier" used in the sprintf function. The format specifier is always prefixed with a percentage sign (%). Please note that some of the format specifiers may not work with your specific MCU. I seemed to get this to work fine on an ESP32, but on an Arduino, the float, scientific and big number sections were a bit hit and miss.

|  |  |
| --- | --- |
|  | /\*========================================================================== |
|  | \* Project: sprintf functionality |
|  | \* Author: Scott C |
|  | \* Date created: 06 May 2019 |
|  | \* Arduino IDE version: 1.8.5 |
|  | \* Operating System: Windows 10 Pro |
|  | \* Tutorial Link: https://arduinobasics.blogspot.com/2019/05/sprintf-function.html |
|  | \* |
|  | \* Acknowledgements: |
|  | \* The following resource was a key element of this tutorial: http://www.cplusplus.com/reference/cstdio/printf/ |
|  | \* Another useful resource can be found here: https://en.wikipedia.org/wiki/C\_data\_types |
|  | \* |
|  | \*------------------------------------------------------------------------------ |
|  | \* Code Explanation |
|  | \* ----------------------------------------------------------------------------- |
|  | \* Begin serial communication at a baud rate of 9600 |
|  | \* Wait until Serial communication has established before continuing |
|  | \* |
|  | \* The sprintf function will write the formatting string and the variables into the "data" character array. |
|  | \* You provide a formatting string template, that contains placeholders for variables that you plan to insert. |
|  | \* These placeholders have a percentage sign (%) prefix. Eg. %s, %d, %f etc. |
|  | \* The number of placeholders must match the number of variables. |
|  | \* The variables are inserted at the placeholder position, and formatted based on the type of placeholder used. |
|  | \* %d = signed integer %f = floating point number |
|  | \* %s = string %.1f = float to 1 decimal place |
|  | \* %c = character %.3f = float to 3 decimal places |
|  | \* %e = scientific notation %g = shortest representation of %e or %f |
|  | \* %u = unsigned integer %o = unsigned octal |
|  | \* %x = unsigned hex (lowercase) %X = unsigned hex (uppercase) |
|  | \* %hd = short int %ld = long int |
|  | \* %lld = long long int |
|  | \* ============================================================================= \*/ |
|  |  |
|  | char data[100]; |
|  | char\* myName = "ArduinoBasics"; |
|  | char\* myBlog = "https://arduinobasics.blogspot.com"; |
|  | int year = 2019; |
|  | int num = 49; |
|  | char myChar = 'a'; |
|  | float myFloat = 99.9911; |
|  | long long int vLong = 1234567890123456789LL; |
|  |  |
|  |  |
|  | void setup() { |
|  | Serial.begin(9600); |
|  | while (!Serial) { |
|  | delay(10); |
|  | } |
|  |  |
|  | sprintf(data, "Tutorial by: %s", myName); |
|  | Serial.println(data); |
|  |  |
|  | sprintf(data, "Year: %u, Blog Link: %s",year,myBlog); |
|  | Serial.println(data); |
|  |  |
|  | Serial.println(); |
|  |  |
|  | sprintf(data, "Character: %c , %c", num, myChar); |
|  | Serial.println(data); |
|  |  |
|  | sprintf(data, "Integer (d): %d , Width(8d): %8d , Zero Prefix Width(07d): %07d", num, num, num); |
|  | Serial.println(data); |
|  |  |
|  | sprintf(data, "Floats f: %f , .1f: %.1f , .3f: %.3f", myFloat, myFloat, myFloat); |
|  | Serial.println(data); |
|  |  |
|  | sprintf(data, "Scientific e: %e , g: %g", myFloat , myFloat); |
|  | Serial.println(data); |
|  |  |
|  | sprintf(data, "Radices u: %u , #o: %#o , x: %x , #x: %#x ", num, num, num, num); |
|  | Serial.println(data); |
|  |  |
|  | sprintf(data, "Big Number lld: [%lld + 1 = %lld]", vLong, vLong+1); |
|  | Serial.println(data); |
|  | } |
|  |  |
|  | void loop() { |
|  | //do nothing |
|  | } |

[**view raw**](https://gist.github.com/ArduinoBasics/6517378fc2ef9e851773837f301b1d55/raw/7638c7c5600107a667b64cf2337a4964bbd2775a/sprintf_functionality.ino)[**sprintf\_functionality.ino**](https://gist.github.com/ArduinoBasics/6517378fc2ef9e851773837f301b1d55#file-sprintf_functionality-ino) hosted with ❤ by **[GitHub](https://github.com/)**

**Serial Monitor Output**

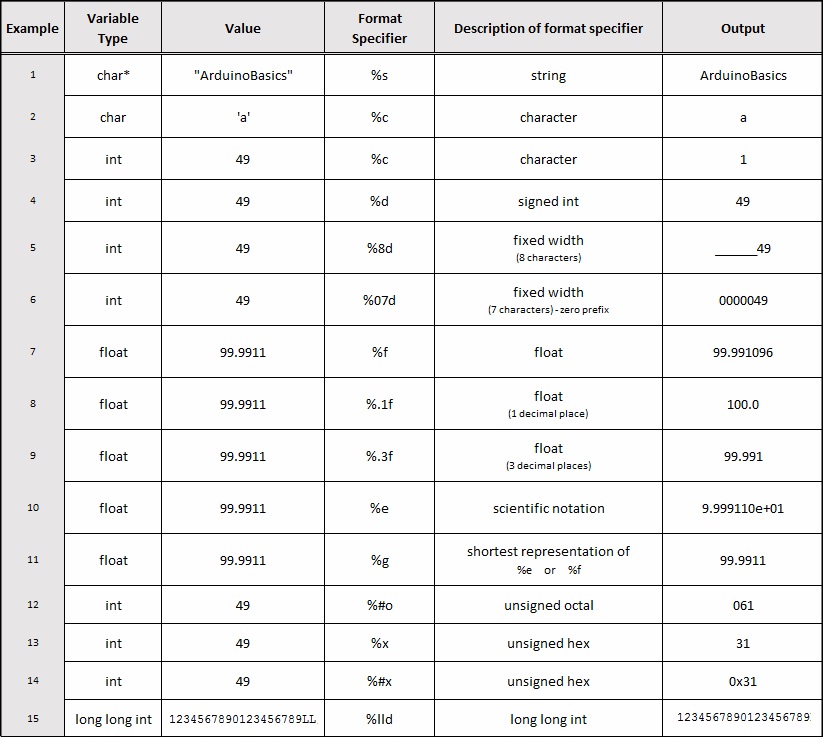
1. Upload the code to the Arduino.
2. Open the Serial monitor in the Arduino IDE (Ctrl+Shift+M).
3. Ensure that you have set the baud rate in the Serial monitor to 9600.
4. You should see the following output:



The sprintf function requires that you have a character array to store the output. In the example code above, the output is stored in the "data" character array. It also requires a template that tells the function where to insert the variables. As you can see from the table below, the variables will be formatted based on the format specifier used. The format specifier can be quite useful for numeric conversions. Eg. decimal to hex conversions.

**Format Specifiers**

Some of the different format specifiers that can be used with the sprinf function are listed below.



**Conclusion**

Now that you know all about the sprintf function, I hope it will inspire you to use it in your own projects. Please let me know in the comments below how you use the sprintf function, and whether there was anything that you feel I failed to mention in this tutorial.  
Happy Coding !!