Strings

In other lanugages, such as Java, there are dedicated *String* data types, but in C the convention is to use an array of chars, which does essentially the exact same thing. A C string is any array of chars followed by the null character, `\0`. The null character is also known as the string terminator (see figure 1). When you go to run your code the C compiler needs to know where every string ends, or terminates. To do this, the compiler parses your program and everywhere it finds closing quotation marks it insterts a character known as the **string terminator**. The string terminator is simply `\0` and takes up one additional byte of memory. This means that if you create a String variable to store your name, and if your name is 8 characters long, the name variable will actually take up 9 bytes of memory. This section outlines a number of different ways that strings can be created in C, as well as looking at how to manipulate stings with functions from the `string.h` library.

Array of Chars

The long way of creating a string is one character at a time, so if you really want to you can do the following:

```
char str1[20] = {'H','e','l','l','o',' ','W','o','r','l','d', '!'};
```

Note that when working with chars that each element of the array has to be inside single quotes. A more concise way to acheive exactly the same thing is by enclosing your message in double quotes like so:

```
char str2[20] = "Hello World!";
```

Try creating a program that includes `str1` and `str2` and prints the both to the console.

string.h

Since the original C language doesn't contain functions for working with strings, much of this functionality is provided by the `string.h` library.

Copying Strings

To overwrite an existing string use the `strcpy()` function. This function takes two **arguments** or parameters. Remember, the `strcpy()` function can be used to overwrite *any* string, so the first thing you need to tell is is what string to replace (overwrite), then you need to tell it what to replace it with i.e. the string you wish to duplicate.

What do you expect the output of the following gist will be?

Download Code

Measuring Strings

To find the length of a string simply use `strlen()` function. This function only accepts one argument, the string you want to measure the length of, and it returns an integer value with the length of the string.

Joining Strings

In many areas, particularly when working with databases, you frequently have to put two (or more) strings of text together. This joining operation is known as **concatenation**. `strcat()`

Comparing Strings

You can check if two strings are identical or not by using the string compare function `strcmp()`. As you can probably imageine, this function requires two arguments - the two strings you want to compare. If, for example, you want to compare *str1* and *str2* there are three possile outcomes. They're either identical, or string 1 might be smaller than string 2, or string 1 might be bigger than string 2. The `strcmp(str1, str2)` function returns 0 if both strings are identical, a negative number if *str1* is less than *str2* or a positive number if *str1* is greater than *str2*.

Coding challenge

Create a simple password app. Your code should should include a user-configurable access key. You should also make sure that your user can not enter a password longer than 20 characters.

```
char key[20] = "YourPassword";
```

and some way to check if the users password is the same as your stored key.