A list of characters is a string.

We use arrays of type char:

```c
char s[] = "My string!";
```

<table>
<thead>
<tr>
<th>'M'</th>
<th>'y'</th>
<th>' '</th>
<th>'s'</th>
<th>'t'</th>
<th>'r'</th>
<th>'i'</th>
<th>'n'</th>
<th>'g'</th>
<th>'!'</th>
<th>'\0'</th>
</tr>
</thead>
<tbody>
<tr>
<td>77</td>
<td>121</td>
<td>32</td>
<td>115</td>
<td>116</td>
<td>114</td>
<td>105</td>
<td>110</td>
<td>103</td>
<td>33</td>
<td>0</td>
</tr>
</tbody>
</table>

The final '\0' character signifies the end of the string.

Note: there may be more entries in the array, but they will be ignored!
Each `char` is essentially a number, but gets translated into a letter with the ASCII table.

http://www.asciitable.com

Important for homework: upper-case letters ’A’ through ’Z’ appear as contiguous numbers, in order.

Lower-case letters ’a’ through ’z’ appear as contiguous numbers, in order.

You can treat a `char` type (including string index) and `char` literal (such as ’a’) as numbers!

- You can use comparisons: `==`  `!=`  `<=`  `>=`
- You can use operations: `+`  `-`  `/`  `*`
Dealing with Literal Strings

- We have seen string literals before, using double quotes ("")
- Examples:
  - printf("Hello world!");
  - scanf("%lf", val);
- These strings cannot be changed after compile time, since they are never stored into an array.
You can use static arrays to store strings.

```c
char string1[] = "Length determined by string";
char string2[100] = "Length determined by 100";
```

Or, you can use dynamic arrays.

```c
char* string3 = malloc(sizeOfString3);
```

Then, you can output strings with `printf`:

```c
printf("My string is:  %s\n",string2);
printf("My string is:  %s\n",string3);
```
Pulling Characters

- We can access individual characters by treating the string as an array.

```c
char string[] = "What is the word?";
printf("Starts with %c\n", string[0]);
```
There are a lot of built-in functions in `string.h`
#include <string.h>

Key pattern: first parameter will store the result.
- `strcpy(char*, char*)`
- `strcat(char*, char*)`
- `strncpy(char*, char*, int)`
- `strncat(char*, char*, int)`