

Computer Science & Engineering 150A

Problem Solving Using Computers – Laboratory

Lecture 09 – Strings

Derrick Stolee

Spring 2009

- A list of characters is a *string*.
- We use arrays of type `char`:

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

'M'	'y'	' '	's'	't'	'r'	'i'	'n'	'g'	'!'	'\0'
77	121	32	115	116	114	105	110	103	33	0

- 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: `+` `-` `/` `*`

- 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.

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

- Or, you can use dynamic arrays.

```
char* string3 = malloc(sizeofString3);
```

- Then, you can output strings with printf:

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

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

```
char string[] = "What is the word?";  
printf("Starts with %c\n", string[0]);
```

- There are a lot of built-int 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)`
- More available in textbook. See Homework 4.