Strings in C

What is a "string"?

- A "string" is just a vector of ASCII characters
  - Followed by a "null terminator" – a byte with the value 0x00

char str[14]="This a string";

{'T', 'h', 'i', 's', ' ', 'a', ' ', 's', 't' ,'r' ,'i' ,'n' ,'g', x00}

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13
ASCII	T	h	i	s		a		s	t	r	i	n	g	
Hex	x54	x68	x69	x73	x20	x61	x20	x73	x74	x72	x69	x6e	x67	x00

Printf substitutes string for %s

```
char str[14]="This a string";
printf("Variable str contains: %s : and no more\n",str);
```

Variable str contains: This a string : and no more

Empty String

char str[14]=“This a string”;
str[0]=x00;
printf(“Variable str contains: %s : and no more\n”,str);

Variable str contains: : and no more

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13
ASCII		h	i	s		a		s	t	r	i	n	g	
Hex	x00	x68	x69	x73	x20	x61	x20	x73	x74	x72	x69	x6e	x67	x00

Standard Library String Functions

```
#include <string.h>
char str[18]="This a string";
printf("Size of str buffer: %d\n",sizeof(str));
printf("Length of str string: %d\n",strlen(str));
```

```
Size of str buffer: 18
Length of str string: 13
```

strlen(char str[]);

- Returns the number of bytes up to (but not including) the null terminator of the argument string.

- Because we start indexes at zero:

```
str[strlen(str)]==x00 // ALWAYS TRUE!
```

```
char empty[20]="";
printf("Length of empty: %d\n",strlen(empty);
Length of empty: 0
```

strcpy(char to[],char from[])

- Copies “from” string to “to” string
- ASSUMES to is large enough to hold strlen(from)

```
char buf[100]=“Old string“;
char new[20]=“Newer string”;
strcpy(buf,new);
printf(“Variable buf contains: %s : and no more\n”,buf);
```

Variable buf contains: Newer string : and no more

strcat(char start[],char tail[])

- Copies “tail” string at the end of “start” string
- ASSUMES start is large enough to hold both start and tail

```
char start[100]=“Beginning “;
char end[20]=“of a test.”;
strcat(start,end);
printf(“Variable start contains: %s : and no more\n”,start);
```

Variable start contains: Beginning of a test. : and no more

strncat(char start[],char tail[],int n)

- Copies up to n bytes of “tail” string at the end of “start” string
- ASSUMES start is big enough to hold both start and n bytes of tail
- Safer than strcat

```
char start[100]=“Beginning “;
char end[20]=“of a test.”;
strncat(start,end,6);
printf(“Variable start contains: %s : and no more\n”,start);
```

Variable start contains: Beginning of a t : and no more

To be totally safe….

```
strncat(start,end,sizeof(start)-strlen(start));
```

strcmp(char a[], char b[])

- Compares the string in "a" to the string in "b"
  - If a<b, returns a number less than zero
  - If a==b, returns zero
  - If a>b, returns a number greater than zero
- Cannot compare strings with ==, <, >, <=, etc. operators!
  - Can compare CHARACTERS with ==, <, …

```
if (0==strcmp(name,"Tom")) printf("Hi Tom…");
```

strncmp(char a[], char b[],int n)

- Compares the string in "a" to the string in "b" for up to n characters
  - If a<b, returns a number less than zero
  - If a==b, returns zero
  - If a>b, returns a number greater than zero
- Cannot compare strings with ==, <, >, <=, etc. operators!
  - Can compare CHARACTERS with ==, <, …

```
if (0==strncmp(name,"Tom",3))
        printf("Name starts with Tom…");
```

Resources

- Programming in C, Chapter 9
- Wikipedia C String Handling
  https://en.wikipedia.org/wiki/C_string_handling
- C String Tutorial :
  http://www.tutorialspoint.com/cprogramming/c_strings.htm