

# Strings

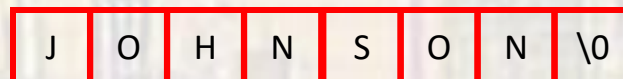
Last updated 12/6/22

These slides introduce strings in C

# Strings

- Strings in C
  - A string is a data structure used to treat a series of characters as a single unit
  - C strings are “delimited” strings
    - Use a delimiter to indicate the end of the string
    - The name of the string is a pointer to the first character in the string – just like an array
    - C uses the ASCII null character as its delimiter ‘\0’

myString

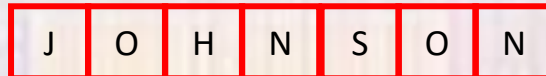


# Strings

- Strings - memory

- An array

myArray



- A string

myString



# Strings

- String Literal (string constant)
  - Characters enclosed in double quotes

“hello world”

“my string literal”

character

‘a’

a

string

“a”

a \0

empty string

“”

note: no space

\0

# Strings

- String Literal (string constant)
  - Characters enclosed in double quotes
  - We can access the individual elements of a string literal

“hello world”

“hello world”[3] → l

“hello world”[6] → w

“hello world”[11] → \0

# Strings

- Declaration

```
char myString[12];
```

- String size must be 1 byte larger than the largest allowed value (to hold the delimiter)
  - This string can hold 11 characters

# Strings

- Initialization

```
char myString[12] = "hello world";
```

```
char myString[] = "hello world";
```

Size is set by the initial value (+1)

```
char myString[12] = {'h', 'e', 'l', 'l', 'o', ' ',  
                    'w', 'o', 'r', 'l', 'd', '\0'};
```

# Strings

- Initialization

```
char myString[12] = "hello world";
```

myString



```
char myString[12] = "hello";
```

myString

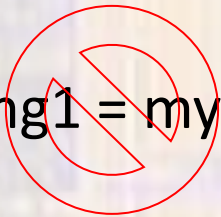




# Strings

- Assignment
  - Just like arrays, **strings cannot be assigned as a whole entity**
  - Must assign element by element

```
mystring1 = mystring2;
```



# Strings

- Strings in C
  - Both Array notation and Pointer notation work

```
/*  
 * strings.c  
 *  
 * Created on: Jan 23, 2018  
 * Author: johnsontimoj  
 */  
  
#include <stdio.h>  
  
#define N 5  
  
int main(void){  
    setbuf(stdout, NULL); // disable buffering  
  
    // local variables  
    char my_string[17] = "this is a string";  
    char* str_ptr;  
    str_ptr = my_string;  
  
    // print some values  
    printf("my_string[5] is %c\n", my_string[5]);  
    printf("char at str_ptr + 6 is %c", *(str_ptr + 6));  
  
    return 0;  
} // end main
```

Array notation

pointer notation

```
<terminated> (exit value: 0) Class_C  
my_string[5] is i  
char at str_ptr + 6 is s
```

Name	Type	Value	Location
my_string	char [17]	0x61ff0b	0x61ff0b
my_string[0]	char	116 't'	0x61ff0b
my_string[1]	char	104 'h'	0x61ff0c
my_string[2]	char	105 'i'	0x61ff0d
my_string[3]	char	115 's'	0x61ff0e
my_string[4]	char	32 ' '	0x61ff0f
my_string[5]	char	105 'i'	0x61ff10
my_string[6]	char	115 's'	0x61ff11
my_string[7]	char	32 ' '	0x61ff12
my_string[8]	char	97 'a'	0x61ff13
my_string[9]	char	32 ' '	0x61ff14
my_string[10]	char	115 's'	0x61ff15
my_string[11]	char	116 't'	0x61ff16
my_string[12]	char	114 'r'	0x61ff17
my_string[13]	char	105 'i'	0x61ff18
my_string[14]	char	110 'n'	0x61ff19
my_string[15]	char	103 'g'	0x61ff1a
my_string[16]	char	0 '\0'	0x61ff1b
str_ptr	char *	0x61ff0b 'this is a str...	0x61ffc
*str_ptr	char	116 't'	0x61ff0b

Value of my\_string (ptr)

Characters – 1B each

ptr to my\_string[0]

Terminator '\0'

# Strings

- String Functions
  - There is a large collection of string functions included in C distributions