String Basics

Last updated 1/14/19

String Basics

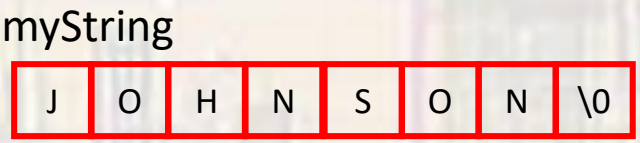
- Built-in Character Functions
  - Testing
    - isalpha()
      - True if letter, False if not
    - isalnum()
      - True if alphanumeric
    - isdigit()
      - True if 0-9
    - islower()
      - True if lower case
    - isprint()
      - True if printable
    - ispunct()
      - True if printable and not alphanumeric
    - isupper()
      - True if uppercase
    - isspace()
      - True if space, tab, newline, vertical tab

String Basics

- Built-in Character Functions
  - Conversion
    - toupper()
      - Converts character to upper case
    - tolower()
      - Converts character to lower case

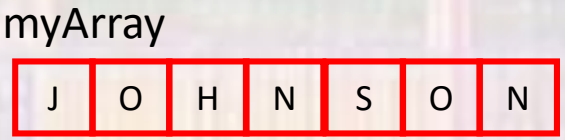
String Basics

- 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’

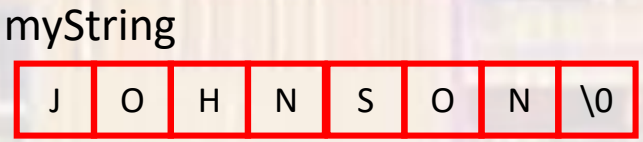


String Basics

- Strings in C++ - memory
  - An array



  - A string



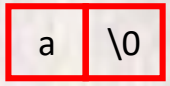
String Basics

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

“hello world”
“my string literal”

character ‘a’

string “a”



empty string “” note: no space

String Basics

- Strings in C++
  - 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”[10] → \0

String Basics

- Strings in C++
  - Declaration

    char myString[12];

  - String size must be 1 element larger than the largest allowed value (to hold the delimiter)

String Basics

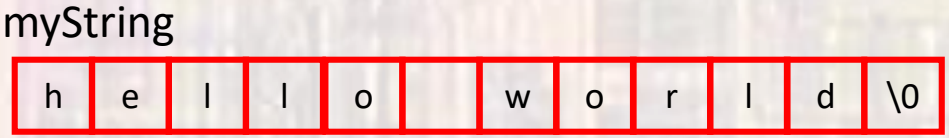
- Strings in C++
  - Initialization

```
char myString[12] = “hello world”;

char myString[] = “hello world”;

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

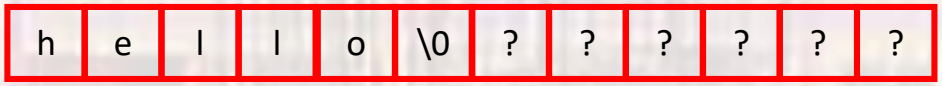
String Basics

- Strings in C++
  - Initialization

char myString[12] = “hello world”;



char myString[12] = “hello”;



String Basics

- Strings in C++
  - Assignment
    - Just like arrays, strings cannot be assigned as a whole entity
    - Must assign element by element
    - There is however a large collection of string functions included in C++ distributions

String Basics

- String Functions
  - printf()

```
printf("my string is: %s", myString);
```

  - scanf()

```
char month[10];                 // create string
scanf("%9s", month);            // read in 9 characters
                                // for the string called
                                // month

fflush(stdin);                  // required to remove
                                // any extra characters
                                // and the newline
```

** if we read in more characters than the string can hold we will overwrite unrelated data

String Basics

- String Functions
  - Line to string – converts a line (up to newline) to a string
  - gets(char* stringPtr)

```
char myString[81];          // standard 80 character line
…
gets(myString);             // read one line of input
```

String Basics

- String Functions
  - String to line – converts a string line of output (including the newline)
  - puts(const char* stringPtr)

```
…
puts(myString);          // output 1 line with value
                         // myString
```

String Basics

- String Functions - #include <string.h>
  - String length – outputs the length of a string excluding the null character
  - int strlen(const char* string)

```
…
foo = strlen(myString);
```

String Basics

- String Functions - #include <string.h>
  - String copy – copy one string to another
  - char* strcopy(char* toStr, const char* fromStr)
    - returns the address of toStr

    …

    strcopy(string2, string1);

    NO Boundary or Size checking is done

  - Use strncopy
  - char* strncopy(char* toStr, const char* fromStr, int size)

String Basics

- String Functions - #include <string.h>
  - String compare – compare 2 strings
  - int strcmp(const char* str1, const char* str2)
    - returns 0 if equal
    - returns <0 if str1 < str2
    - returns >0 if str1 > str2

```
if(strcmp(mystr1, mystr2) == 0)
    …
```

  - int strncmp(const char* str1, const char* str2, int size)
  - Compares the first N elements

String Basics

- String Functions - #include <string.h>
  - String concatenation – concatenate 2 strings
  - char* strcat(char* str1, const char* str2)
    - returns the address of toStr

    …

    strcat(stringA, stringB); // result in stringA

    NO Boundary or Size checking is done

  - Use strncat
  - char* strncat(char* str1, const char* str2, int size)

String Basics

- String/Numeric Conversions #include <csdtlib.h>
  - atoi()
    - Converts a string to an integer
      - atoi("1234") returns an int, 1234
  - atof()
    - Converts a string to an float
      - atoi("12.34") returns a float, 12.34
  - itoa(value, string, base)
    - Converts an integer to a string
      - Value – integer to convert
      - String – pointer to save string
      - Base – base for the conversion, 8, 10, 16, …