C++ for Science and Engineering COSC3000/6000

2018 Spring Semester

Part XII
Strings and Vectors

1 C-Strings

- C-strings can be used to represent strings of characters
  - C-strings are stored as arrays of characters
  - C-strings use the null character '\0' to end a string
    - The Null character is a single character
  - To declare a C-string variable, declare an array of characters:

```
char s[11];
```

1.1 C-string Details

- Declaring a C-string as char **s[10]** creates space for only **nine** characters
  - The null character terminator requires one space
- A C-string variable does not need a size variable
  - The null character immediately follows the last character of the string
- Example:

s[0]	s[1]	s[2]	s[3]	s[4]	s[5]	s[6]	s[7]	s[8]	s[9]
C	O	S	C	3	0	0	0	\0	?

- To declare a C-string variable, use the syntax:

```
char Array_name[ Maximum_C_String_Size + 1];
```

  - + 1 reserves the additional character needed by **'\0'**

1.2 Initializing a C-string

- To initialize a C-string during declaration:

```
char my_message[20] = "Hi COSC3000!";
```

  - The null character **'\0'** is added for you

- Another alternative:
```
char short_string[ ] = "abc";
```
but not this:
```
char short_string[ ] = {'a', 'b', 'c'};
```
this works:
```
char short_string[ ] = {'a', 'b', 'c','\0'};
```

1.3 Assignment With C-strings

- This statement is illegal:
```
a_string = "Hello";
```

  - This is an assignment statement, not an initialization
  - **The assignment operator does not work with C-strings**

1.3.1 Assignment of C-strings

- A common method to assign a value to a C-string variable is to use **strcpy**, defined in the **cstring** library
  - Example:
```
#include <cstring>
:
:
char a_string[11];
strcpy (a_string, "Hello");
```
Places **"Hello"** followed by the null character in **a_string**

1.3.2 A Problem With strcpy

- **strcpy** can create problems if not used carefully
  - **strcpy** does not check the declared length of the first argument
  - It is possible for **strcpy** to write characters beyond the declared size of the array

1.3.3 A Solution for strcpy

- Many versions of C++ have a safer version of **strcpy** named **strncpy**
  - **strncpy** uses a third argument representing the maximum number of characters to copy
  - Example:
```
char another_string[10];
strncpy(another_string, a_string_variable, 9);
```
This code copies up to 9 characters into **another_string**, leaving one space for **'\0'**

1.4 == Alternative for C-strings

- The = = operator does not work as expected withC-strings
  - The predefined function **strcmp** is used to compareC-string variables
  - Example:
```
if (strcmp(c_string1, c_string2))
        cout << "Strings are not the same.";
else
        cout << "String are the same.";
```
- **strcmp** compares the numeric codes of elements in the C-strings a character at a time

- If the two C-strings are the same, **strcmp** returns 0
  - 0 is interpreted as **false**
- As soon as the characters do not match
  - **strcmp** returns a negative value if the numeric code in the first parameter is less
  - **strcmp** returns a positive value if the numeric code in the second parameter is less
  - Non-zero values are interpreted as **true**

There are many other functions : `http://www.tutorialspoint.com/ansi_c/c_function_references.htm`