THE UNIVERSITY OF WESTERN ONTARIO

DEPARTMENT OF COMPUTER SCIENCE
LONDON CANADA

Software Tools and Systems Programming
(Computer Science 2211a)

LAB 4
The week of October 04, 2020

The purpose of this lab is to demonstrate the use of arrays, specifically character arrays known as strings in C.

PREPERATION:
What is a String?

A string is nothing but a collection of characters in a linear sequence. 'C' always treats a string as a single data even though it contains whitespaces. A single character is defined using **single quote** representation. A string is represented using **double quote marks**.

Example, "Welcome to the world of programming!"

'C' provides standard library <string.h> that contains many functions which can be used to perform complicated string operations easily.

In this tutorial, you will learn-

What is a String?
Declare and initialize a String
String Input: Read a String
String Output: Print/Display a String
fputs() function
puts function

LAB 04:

Declare and initialize a String

A string is a simple array with char as a data type. 'C' language does not directly support string as a data type. Hence, to display a string in 'C', you need to make use of a character array.

The general syntax for declaring a variable as a string is as follows,

```
char string_variable_name [array_size];
```

The classic string declaration can be done as follow:

```
char string_name[string_length] = "string";
```

The size of an array must be defined while declaring a string variable because it used to calculate how many characters are going to be stored inside the string variable. Some valid examples of string declaration are as follows,

char first_name[15]; //declaration of a string variable
char last_name[15];

The above example represents string variables with an array size of 15. This means that the given character array is capable of holding 15 characters at most. The indexing of array begins from 0 hence it will store characters from a 0-14 position.
The C compiler automatically adds a NULL character '\0' to the character array created.

Let's study the initialization of a string variable. Following example demonstrates the initialization of a string variable,

In Unix, if you do not already have a directory named Labs then first create a directory labeled: **`Labs`**
Under that directory, create a new directory labeled: **`lab4`**

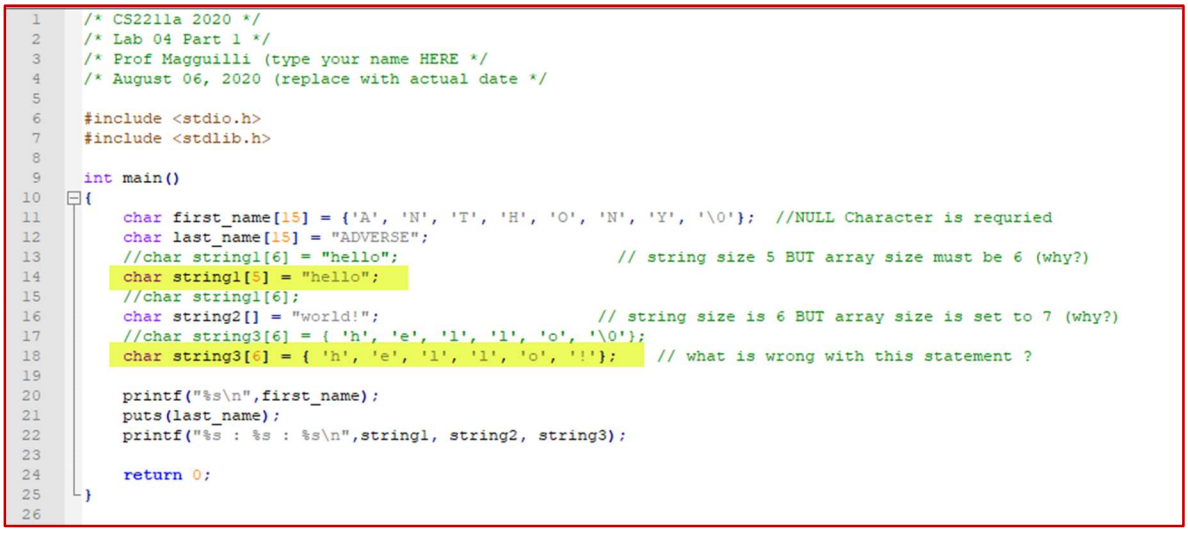
Create a new file using vi or vim and label this: **`lab4A.c`**

Type in the following code EXACTLY as shown:

```c
/* CS211a 2020 */
/* Lab 04 Part 1 */
/* Prof Magguilli (type your name HERE */
/* August 06, 2020 (replace with actual date */

#include <stdio.h>
#include <stdlib.h>

int main()
{
    char first_name[15] = {'A', 'N', 'T', 'H', 'O', 'N', 'Y', '\0'};  //NULL Character is requried
    char last_name[15] = "ADVERSE";
    char string1[6] = "hello";                      // string size 5 BUT array size must be 6 (why?)
    //char string1[5] = "hello";
    //char string1[6];
    char string2[] = "world!";                      // string size is 6 BUT array size is set to 7 (why?)
    char string3[6] = { 'h', 'e', 'l', 'l', 'o', '\0'};
    //char string3[6] = { 'h', 'e', 'l', 'l', 'o', '!'};    // what is wrong with this statement ?

    printf("%s\n",first_name);
    puts(last_name);
    printf("%s : %s : %s\n",string1, string2, string3);

    return 0;
}
```

Compile the program (naming the executable **`lab4A`**) and run.
The results are as expected.

Next, change the code slightly as shown below (new lines are highlighted in yellow).



```
1  /* CS211a 2020 */
2  /* Lab 04 Part 1 */
3  /* Prof Magguilli (type your name HERE */
4  /* August 06, 2020 (replace with actual date */
5
6  #include <stdio.h>
7  #include <stdlib.h>
8
9  int main()
10 {
11     char first_name[15] = {'A', 'N', 'T', 'H', 'O', 'N', 'Y', '\0'};  //NULL Character is requried
12     char last_name[15] = "ADVERSE";
13     //char string1[6] = "hello";                    // string size 5 BUT array size must be 6 (why?)
14     char string1[5] = "hello";
15     //char string1[6];
16     char string2[] = "world!";                    // string size is 6 BUT array size is set to 7 (why?)
17     //char string3[6] = { 'h', 'e', 'l', 'l', 'o', '\0'};
18     char string3[6] = { 'h', 'e', 'l', 'l', 'o', '!'};    // what is wrong with this statement ?
19
20     printf("%s\n",first_name);
21     puts(last_name);
22     printf("%s : %s : %s\n",string1, string2, string3);
23
24     return 0;
25 }
26
```

This is the code to enter:

```
char first_name[15] = {'A','N','T','H','O','N','Y','\0'};
char last_name[15] = "ADVERSE";
//char string1 [6] = "hello";
char string1 [6];
char string2 [ ] = "world";
//char string3[6] = {'h', 'e', 'l', 'l', 'o', '\0'} ;
char string3[6] = {'h', 'e', 'l', 'l', 'o','!'} ;
```

comment out lines 13 and 17 and uncomment lines 14 and 18.

Compile the new code and run. Notice how the output changed.
Why did these two small edits change the output?
This is a rhetorical question – you do NOT have to type the answer anywhere in the lab.
BUT! Be prepared to answer this in a quiz or exam.

Lastly, repeat this process, but now comment out line 14 and uncomment line 15.

PREPERATION:
For the remainder of the lab, use the supplied file lab4B.c included with these instructions.

Copy the file into your: **`lab4`** directory that you have created.

String Input: Read a String

When writing interactive programs which ask the user for input, C provides the scanf(), gets(), and fgets() functions to find a line of text entered from the user.

When we use scanf() to read, we use the "%s" format specifier without using the "&" to access the variable address because an array name acts as a pointer.

In the main.c file entered in vi (or vim), remove the comments from line 22 : `readString();`

```
int readString() {
   char name[10];
   int age;
   printf("Enter your first name and age: \n");
   scanf("%s %d", name, &age); }
   printf("You entered: %s %d",name,age);
   return 0;
}
```

Output:
Compile and run the program.
Enter your first name only and your age and view the result.

```
Enter your first name and age:
<type in your name> <type in your age>
```

The problem with the scanf function is that it never reads an entire string. It will halt the reading process as soon as whitespace, form feed, vertical tab, newline or a carriage return occurs. Suppose we give input as "Max Maggs" then the scanf function will never read an entire string as a whitespace character occurs between the two names. The scanf function will only read Max.

In order to read a string contains spaces, we use the gets() function. Gets ignores the whitespaces. It stops reading when a newline is reached (the Enter key is pressed).

For example:
In the main.c file entered in Vi (or vim), replace the comment tag for line 22 and remove the comments from line 23 : `readFullString();`

```
int readFullString () {
   char full_name[25];
   printf("Enter your full name: ");
   gets(full_name);
   printf("My full name is %s ",full_name);
   return 0;
}
```

Output:
Compile and run the program.
Enter your first name only and your age and view the result.

```
Enter your full name:
<type in your full [first and last name]>
```

The problem with the scanf function is that it never reads an entire string. It will halt the reading

String Output: Print/Display a String

The standard printf function is used for printing or displaying a string on an output device. The format specifier used is %s

Example,

```
printf("%s", name);
```

String output is done with the fputs() and printf() functions.

fputs() function

The fputs() needs the name of the string and a pointer to where you want to display the text. We use stdout which refers to the standard output in order to print to the screen.

For example:
In the main.c file entered in Vi (or vim), replace the comment tag for line 23 and remove the comments from line 24 : `fputsFunc ();`

```
int fputsFunc ()
{char town[40];
  printf("Enter the city/town you were born in: ");
  gets(town);
  fputs(town, stdout);
  return 0;
}
```

Output:

```
Enter the city/town you were born in: New York
New York
```

puts function

The puts function prints the string on an output device and moves the cursor back to the first position. A puts function can be used in the following way:

In the main.c file entered in Vi (or vim), replace the comment tag for line 24 and remove the comments from line 25 : `putsFunc ();`

```
int putsFunc () {
  char name[15];
  gets(name);       //reads a string
  puts(name);       //displays a string
  return 0;
}
```

Summary

- A string is a sequence of characters stored in a character array.
- A string is a text enclosed in double quotation marks.
- A character such as 'd' is not a string and it is indicated by single quotation marks.
- 'C' provides standard library functions to manipulate strings in a program. String manipulators are stored in <string.h> header file.
- A string must be declared or initialized before using into a program.
- There are different input and output string functions, each one among them has its features.
- Don't forget to include the string library to work with its functions
- We can manipulate different strings by defining a string array.

FINISH:

1. Type the following to begin recording your session in a file called *yourUserName*_lab4**.output**

script YourUserName_lab4.script

- (using your actual user name).

2. Display the current date and time using the appropriate command

3. Display your username using the appropriate command

4. Display the contents of the current working directory using the 'l' switch (lower case L)

5. Compile both programs again just using the last version
   (do NOT do all the different versions – just show the program compiles and runs)

6. Display to the screen the lab4A.c main file.

7. Display to the screen the lab4B.c main file

8. Run the program: *lab4A*.

9. Run the program *lab4B*.

10. Type **`exit`** to stop your screen capture session.

Submission Instructions:

Required Coding Standards

All code is to be indented correctly.
Comments at the very beginning (top – first lines) of each of the .c files must be:

```
/* CS2211a 2020 */
/* Lab 04 */
/* your name */
/* your student number */
/* your UWO User Name */
/* Date Completed */
```

Submit via the CS2211 OWL Web Site the following four (4) files along with any other files (i.e. executables etc.) in OWL inside your tar file:

lab4a.c
Lab4b.c
*YourUserName*_lab4.script
CS2211a Lab Submission Form (see below on how to submit this form)

(*yourUserName* - example: assume my UWO email is kdoit373@uwo.ca
i.e. if my email is – **kdoit373@uwo.ca** then my user name will be – **kdoit373**
So, my UWO User Name is: **kdoit373** and this lab is **lab4**
therefore, one of the file names that is to be used for submission is:
kdoit373_lab4.script

It is the student's responsibility to ensure the work was submitted and posted in OWL. OWL replies with a summation verification email (every time).

Any labs **not** submitted correctly will **not** be graded.

Reminder how to submit Labs:

- In your Gaul account, go to directory *~/courses/cs2211a/Labs/lab4*

• From your home computer, access course website and download *"CS2211a Lab submission form"* from the Course Resources section of OWL.
Submit the *CS2211a Lab Submission Form*

• **To accomplish the submission of the** *CS2211a Lab Submission Form***:**

• - First go to the <Course Resource> section.

• - Look for the link titled "CS2211a Lab Submission Form"

• - Make a copy of that form so you can edit (change) the contents of the form.

• - Fill in the information on that form by replacing the prompts with your actual data (i.e. for Student Last Name - please enter your last name, etc.)

• Include the text regarding the declarations (found under the paragraph:
By submitting this form, I declare that:)

• If you have submitted a self-reported absence academic consideration for this Lab, you must include the required information. (contact us if you are unsure what information is required).

• - Save a copy of that form.

• Name (or rename) that form to **Lab_SubmissionForm.txt**

- Save this form as a text file or as **PDF** (most word processors have this option).
  - if you are unsure how to convert a file to a PDF please contact your TA
  - you are NOT to submit an MS Word or any word processed file.
  - ONLY a .txt or .pdf

- Use **sftp** (or any file transfer service like FileZilla or WINSCP) to upload the completed version of the *"CS2211a Lab submission form"* that you entitled Lab_SubmissionForm.txt from your home computer to the directory *~/courses/cs2211a/Labs/lab4* in you Gaul account.

- Use **sftp** (or any file transfer service like FileZilla or WINSCP) to upload the pdf or text version of the *"CS2211a lab submission form"* that you have entitled Lab_SubmissionForm.txt ***from*** your home computer **to** the directory *~/courses/cs2211a/Labs/lab4* in you Gaul account.

- In your Gaul account, go to directory *~/courses/cs2211a/Labs/lab4*

- Use **tar -czvf tar *file name* dir *name*** to create a compressed tar file containing all the files and directories under *~/courses/cs2211a/Labs/lab4*

  - In your Gaul account, goto directory *~/courses/cs2211a/Labs*

- Use **tar -czvf YourUserName_lab4.tar.gz lab4** to create a compressed tar file *YourUserName_lab4.tar.gz* containing all files and directories under *~/courses/cs2211a/Labs/lab4*. **YourUserName** should be your UWO email user name or your Gaul account user name.

- Go to *~/tmp* directory and open and restore the tar file with **tar -xvf tar file name**. The purpose of this step is to make sure that you did not make mistake in the previous step in creating the tar file.
  - Copy *YourUserName lab4.tar.gz* to *~/tmp*.
  - Go to directory *~/tmp*.
  - Use **tar -xvf YourUserName _lab4.tar.gz** to restore *lab4* under *~/tmp*
  - Check if you now have two identical directories *~/courses/cs2211a/Labs/lab4* and *~/tmp/lab4*.
    In each directory, you should have all the files you are submit
- Use **sftp** to download tar file *YourUserName lab4.tar.gz* from your Gaul account to your home computer.
- From your home computer, you are now ready to submit this file, *YourUserName lab4.tar.gz*, for lab 4 through OWL.
- NOTE: It is your responsibility to check and ***then re-check*** that all the files are in the compressed tar file you submit to OWL.

Please let us know if any of these instructions (and if possible, which exactly) may be unclear, confusing or ambiguous.

PS: remember: do your own work – you will need to know all this for the exam to pass !!!!