Lab 6: Functions and string manipulation

Goals

Understand how to build simple C language functions and use them to process null-terminated text strings.

Background

Strings

- Character strings (or simply strings) are arrays of *char* variables.
- They are used in a C language program to store words or sentences in the English (or some other) language.
- Some programming languages have an actual variable type called string, but C does not.
- The logical end of a C language string is denoted by the presence of a byte consisting of all 0 bits some times called a *NULL byte.*
- The program explicitly must provide storage for the NULL byte.
- Thus, a character array that can hold the string "Hello" must be at least 6 bytes in size: char hello[6];

Functions

- Functions provide a mechanism for partitioning a large, complex program into a collection of small, easily understood components.
- Each function should be designed to solve ONE aspect of the problem at hand.

Function definitions

A C function is comprised of 4 components

- the type of value returned by the function
- the name of the function
- parenthesized declaration of function parameters (values passed to the function by its caller).
- a basic block containing local variable declarations and executable code

```
int sum(
int a,      /* Values to be added */
int b)      /* These are provided by my caller */
{
   int total;

   total = a + b;
   return(total);
}
```

A C function may also be passed an array (technically the address of the first element in the array) as a parameter. In this case it is common to omit the dimension of the array and have it supplied by the caller.

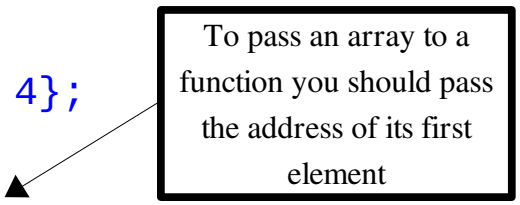
```
int sumtab(
int array[], /* Values to be added */
int num)     /* How many values */
{
   int total = 0;
   int ndx = 0;

   while (ndx < num)
   {
      total = total + array[ndx];
      ndx = ndx + 1;
   }
   return(total);
}
```

A C function is called or invoked by just using its name anywhere a variable or constant might be used in an expression:

```
int main()
{
   int num;
   int table[7] = {1, 2, 3, 4};

   num = sum(4, 5);
   num = num + sumtab(&table[0], 4);
}
```



When the expression is evaluated, the value returned by the function replaces its invocation in the expression. So here, *sum()* will return a 9 and 9 will be assigned to *num.* Then *sumtab()* will return a 10 and so the final value of *num will be 19.*

NOTES:

- Local variable name spaces of different functions are completely disjoint. I can declare a local variable called *total* in *main()* and it will be *completely independent of the total in sum.*

- The correspondence between the formal parameters (a, and b) in *sum()* with the actual arguments (4, and 5) in main() is positional. When this program runs, if we were to include
  fprintf(stdout, "%d %d\n", a,b);
  in *sum()*, we would see a holds the value 4 and b holds 5.

- Therefore, the number and types of the actual arguments used when invoking a function should *exactly match* the formal parameters of the function definition.

Assignment:

Write a program called lab6.c that contains three functions as follows:

1. Create a function `int my_strlen (char s1[])` that will compute and return the length of a string s1. A single while loop should be used to count the characters one at a time. Processing should terminate when the NULL terminator is encountered (s1[ndx] == 0)

2. Create a main() function to test my_strlen().

   a. Define three character arrays v1, v2, and v3 of length 16 and two integer variables l1 and l2.

   b. Read in two strings (maximum length 15) into v1 and v2 from the standard input. Invoke my_strlen() to set l1 and l2 to the lengths of v1 and v2 respectively. Use the format string "%3d - %s \n" to print l1, v1 and then l2, v2 to the standard output.

3. Create a function `void my_strcpy (char s1[], char s2[])` that will copy the contents of s1 into s2. A single while loop should be used to copy the characters one at a time. Copying should end when s1[ndx] == 0, but *care should be taken to ensure that s2 is properly terminated with a NULL.*

4. Augment your main() function so that it will:

   c. Copy the string v1 to v3, and print v3 using the format string "%s".

Turn In Work

Show your TA that you completed the assignment. Then turn in your lab6.c program using the command:

sendlab.101.*section_number* 6 lab6.c