CSCE 155 - C

Lab 09 - Strings

Dr. Chris Bourke

Prior to Lab

Before attending this lab:

1. Read and familiarize yourself with this handout.

2. Review the following free textbook resources:

   - http://en.wikibooks.org/wiki/C_Programming/Strings

Peer Programming Pair-Up

To encourage collaboration and a team environment, labs will be structured in a *pair programming* setup. At the start of each lab, you will be randomly paired up with another student (conflicts such as absences will be dealt with by the lab instructor). One of you will be designated the *driver* and the other the *navigator*.

The navigator will be responsible for reading the instructions and telling the driver what to do next. The driver will be in charge of the keyboard and workstation. Both driver and navigator are responsible for suggesting fixes and solutions together. Neither the navigator nor the driver is "in charge." Beyond your immediate pairing, you are encouraged to help and interact and with other pairs in the lab.

Each week you should alternate: if you were a driver last week, be a navigator next, etc. Resolve any issues (you were both drivers last week) within your pair. Ask the lab instructor to resolve issues only when you cannot come to a consensus.

Because of the peer programming setup of labs, it is absolutely essential that you complete any pre-lab activities and familiarize yourself with the handouts prior to coming to lab. Failure to do so will negatively impact your ability to collaborate and work with others which may mean that you will not be able to complete the lab.

1 Lab Objectives & Topics

At the end of this lab you should be familiar with the following

- Declare and print a string in C

- Manipulate strings in a variety of ways

- Use some basic functions from the `string.h` library

2 Background

Strings are collections of characters. In C, Strings are represented using arrays of char values terminated by a special null-terminating character, `\0`. Because they are arrays, the same precautions must be made when working with strings as with general arrays. The standard string library (`string.h`) provides many convenience functions to manipulate and use strings.

This lab will familiarize you with some of these functions. In particular, you will complete a program that implements a common children's game, horse (also known as hangman). In this game, an English word is chosen at random and its characters hidden. The player takes turns by guessing a letter; each instance (if any) of the guessed letter is revealed. If the user is able to guess the word before a certain number of guessed letters then they win. If they run out of guesses then they lose.

Most of the game mechanics have been implemented for you. However, you will need to complete the game by implementing several functions used by the game to manipulate and compare strings.

3 Activities

Clone the code for this lab from GitHub using the following URL: https://github.com/cbourke/CSCE155-C-Lab09.

3.1 Implementing String Manipulation Functions

1. Navigate to the `src` directory and open `gameFunctions.c` in the editor of your choice.

2. There are several functions already fully implemented in this file. Your task for this lab is to implement the following four functions:

- `initializeBlankString()` - This function should take two arguments: an integer denoting the length of the second argument, which should be a character array. It should return nothing. The function should alter the passed array so that it is filled with underscores, `_` and is a properly terminated string.

- `printWithSpaces()` - This function will take a string as input and print the contents of the string with spaces between each character. (Hint: use the strlen() function to find the size of the passed string). The function should return nothing.

- `revealGuessedLetter()` - This function will take two strings and a character as input. The function should alter the second string in the following way: For every position in the first string that contains the character passed in as the third argument to the function, change the same position in the second string to that character. For example, if the first string is `"dinosaur"` and the second is `"_____"` and the character passed is `a`, then the function should alter the second string so that it becomes `"_____a__"`. You may assume that the strings are of equal length. The function should return a 1 if any letters were changed in the second string and 0 otherwise.

- `checkGuess()` - This function should take two strings as input. If the two strings are equivalent, return a 1 from the function. If they're different, return a 0. There are at least two ways to do this: you may use the `strcmp()` function from the string library or you can iterate over every character in the strings. You may assume that the strings are equal length.

3. Navigate to the `include` directory and open `gameFunctions.h` in the editor of your choice

4. Complete the function prototypes that you implemented in `gameFunctions.c` here.

5. Compile the program using the `make` command and complete the worksheet.

4 Advanced Activity (Optional)

Currently, the game has a strict limitation on the number and length of words a user can enter in the dictionary.txt file. Alter the program so that it can accept any number of words and words of any length from `dictionary.txt` (hint: you'll need to dynamically allocate the memory for the array in `main.c`, among other changes in `gameFunctions.h` and `gameFunctions.c`).