continue, break, system()

arrays/contiguous memory/cstrings

CptS 121 - Program Design and Development

Lab 8: Awesome Arrays

Assigned: Week of October 14, 2024
Due: At the end of the lab session

I. Learner Objectives:

At the conclusion of this programming assignment, participants should be able to:

- Declare arrays in C
- Apply arrays in C to various problems
- Pass arrays into functions
- Initialize arrays using an initializer list
- Construct loops to traverse through arrays

II. Prerequisites:

Before starting this programming assignment, participants should be able to:

- Utilize output parameters and pointers in a C program
- Apply the dereference or indirection C operator
- Discuss advantages and disadvantages of applying pointers in a C program
- Determine how the use of pointers may affect modular design
- Compose iterative statements
- Compose decision statements
- Apply top-down design principles

III. Overview & Requirements:

This lab, along with your TA, will help you navigate through applying arrays in C. We know that arrays are based on the premise of contiguously allocated blocks of memory. This allows for us to use indices to efficiently access items within an array. A single dimensional array may be logical viewed as a single row of multiple cells.

Labs are held in a "closed" environment such that you may ask your TA questions. Please use your TAs knowledge to your advantage. You are required to move at the pace set forth by your TA. Have a great time! Labs are a vital part to your education in CptS 121 so work diligently.

Tasks:

1. Write a program which populates an array with integer values read from a file. We do not know how many integers are in the file, so you must loop until the end of the file is reached. For this problem, you may NOT use the function `feof()`. Instead, use the result returned from `fscanf()` to determine the end-of-file. Recall, we can set the result of `fscanf()` to an integer variable, and check to see if the integer variable is equal to the `EOF` marker. The

program must take the items in the array and reverse them. You may use one array only to solve this problem.

2. Write a function called `remove_whitespace()` that accepts an array of *characters* and the *number* of items in the array as parameters, *removes* all *whitespace* (`' '`) characters from the array, and returns the *number* of whitespace characters removed. We are **NOT** treating the array of characters like a C string.

   For example, if the array contains `['C', 'p', 't', 'S', ' ', '1', '2', '1', ' ', 'i', 's', ' ', 'f', 'u', 'n']`, then the function should remove the whitespace characters. The function must remove the whitespace characters by shifting all characters to the right of each whitespace character, left by one spot in the array. This will overwrite the whitespace characters, resulting in: `['C', 'p', 't', 'S', '1', '2', '1', 'i', 's', 'f', 'u', 'n']`. In this case, the function returns 3. Note: if the array does not contain any whitespace characters, then the array is unchanged, and the function returns 0.

3. Write a program that populates an array with 20 random integers between 1 - 100. The program must traverse through the array and determine how many times each integer was generated. Use a second array of size 101 to keep track of the number of times each integer was generated. Initialize each item in the second array to 0. For each item in the first array, use it as the index into the second array and increment the contents found in the second array at the corresponding index. Note: for this problem we are willing to trade memory efficiency (the second array is mostly unused) for time efficiency. We know indexing into an array is very efficient.

4. Complete the following programming project 9 on p. 451 of your Hanly & Koffman <u>Problem Solving and Program Design in C</u> text. Write an interactive program that plays a game of hangman. Store the word to be guessed in successive elements of an array of individual characters called word (note: this is not a string!). The player must guess the letters belonging to word. The program should terminate when either all letters have been guessed correctly (the player wins) or a specified number of incorrect guesses have been made (the computer wins). Hint: use another array, guessed, to keep track of the solution so far. Initialize all elements of guessed to the '*' symbol. Each time a letter in word is guessed, replace the corresponding '*' in guessed with that letter. string.h!!!!

IV. Submitting Labs:

- You are not required to submit your lab solutions. However, you should keep them in a folder that you may continue to access throughout the semester.

V. Grading Guidelines:

- This lab is worth 10 points. Your lab grade is assigned based on completeness and effort. To receive full credit for the lab you must show up on time, work with a team, complete 2/3 of the problems, and continue to work on the problems until the TA has dismissed you.