



Lab 10: Structs

Assigned: Week of March 31, 2014

Due: At the end of the lab session

I. Learner Objectives:

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

- Utilize arrays of pointers to strings
- Define and declare structures (structs) in C
- Apply the C member dot and pointer operators to structs
- Pass structs between functions

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
- Declare strings in C
- Apply library functions found in <string.h>
- Implement array notation or pointer arithmetic to manipulate strings
- Distinguish between character arrays and strings in C
- Pass arrays into functions
- Initialize arrays using an initializer list
- Construct loops to traverse through arrays

III. Overview & Requirements:

This lab, along with your TA, will help you navigate through applying structures (structs) in C. Recall that a struct in C may be used to describe physical objects in the real world. A single struct is a collection of fields or components that may have several different types. The fields within a struct are contiguous in memory. Also, recall that the dot member operator (.) may be used to access individual fields in a struct. The pointer operator (->) may be used to access fields when a pointer to a struct is present.

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. Please help other students in need when you are finished with a task. You may work in pairs if you wish. However, I encourage you to compose your own solution to each problem. Have a great time! Labs are a vital part to your education in CptS 121 so work diligently.

Tasks:

1. Adapted from Exercise 8.11 in Deitel & Deitel C How to Program. Write a program that uses random

number generation to create sentences. The program should use four arrays of pointers to chars called *article*, *noun*, *verb*, and *preposition*. The program should create a sentence by selecting a word at random from each array in the following order: *article*, *noun*, *verb*, *preposition*, *article*, and *noun*. As each word is picked, it should be concatenated to the previous words in an array large enough to hold the entire sentence. The words should be separated by spaces. When the final sentence is output, it should start with a capital letter and end with an exclamation mark (!). The program should generate 20 such sentences. The arrays should be filled as follows: The *article* array should contain "the", "a", "one", "some", and "any"; the *noun* array should contain "boy", "girl", "dog", "town", and "car"; the *verb* array should contain "drove", "jumped", "ran", "walked", and "skipped"; the *preposition* array should contain "to", "from", "over", "under", and "on". You may add other strings to each one of these arrays if you wish!

2. (a) Write a two-player interactive game to play Tic Tac Toe. The game board for Tic Tac Toe will be represented by a 2-Dimensional array of structs. The underlying game board struct should be represented as follows:

```
typedef struct coordinate
{
    int row;
    int column;
} Coordinate;

typedef struct cell
{
    int occupied; // 1 if an X or O is in this cell; 0 otherwise
    char symbol; // X for one player, O for the other player
    Coordinate location; // A struct defined above, which represents the position of the cell within the
game board
} Cell;
```

The Tic Tac Toe game board consists of a total of $n \times n$ Cells, where n is selected by the user of the game. Your program should randomly select who goes first, X or O. Once a player has been chosen, your program should prompt the player for a position (row and column) in which to draw his/her symbol. Players continue to alternate moves until either a winner has been determined or a "scratch" game occurs. A player wins if his/her symbols align with n in a row diagonally, vertically, or horizontally. A "scratch" game occurs if all cells on the board are occupied and no player aligned n of his/her symbols in a row.


The program should prompt the users to determine if they want to play another game. Keep track of each player's number of wins, losses, and total game played in a struct defined as follows:

```
typedef struct game_info
{
    int wins;
    int losses;
    int total_games_played.
} Game_info;
```


Use functions where appropriate.

(b) Make one of the players the computer. Introduce some basic artificial intelligence so that the computer makes "educated" moves.

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. You should not store your solutions to the local C: drive on the Sloan 353 machines. These files are erased on a daily basis.

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 and continue to work on the problems until the TA has dismissed you.