Lab 11: Pointer Paramters and Struct Pointers

CSci 515 Spring 2015

2015-04-14

Dates:

Due: In Lab, Wednesday April 15, by 4:05 pm (lab end time)

Objectives

- Become familiar with creating and using pointer variables.
- Pass pointers to a function.
- Get practice using structs with pointers.

Description

In this lab you will practice using pointers. We will create some functions that you will pass pointer parameters too in order to perform some tasks.

Perform the following tasks:

1. Create a function called `maxInFirst`. This function will take two float pointer variables as its only parameters. This is a void function, so it doesn't return anything explicitly. The purpose of this function is to determine which of the two values pointed to is larger, and make sure that the largest of the two is at the memory location pointed to by the first parameter. So if the value pointed to by the first parameter is already the largest, then this function does nothing. But if the second pointer value referenced is the larger, then this function needs to swap these two values.
2. Declare a `struct` called `Employee`. The Employee `struct` should have a name field, of type string, and a salary field, of type float.

3. Write a function called `initializeEmployee`. This function will take a pointer to an Employee `struct` as its only parameter. This will be a void function, so it will not return anything explicitly. Have this function prompt the user interactively (inside of the function) for an employee name and a floating point salary. You should initialize the fields of the `struct` that is passed in as a pointer parameter using the values input by the user.

4. In your main function show an example of calling each of your two functions. For the `maxInFirst` function, make sure you demonstrate calling it two times, one time demonstrating when the values need to be swapped, and one time when they done. Also create an Employee variable instance in your main function, and initialize it by calling your `initializeEmployee` function. Display the two fields after the function returns from initializing them.

Example output is shown below. You should prompt for the employee name inside of your function, and display the resulting fields after returning from calling the initialization function.

```
Before calling maxInFirst: f1 = 42.42 f2 = 18.18
After  calling maxInFirst: f1 = 42.42 f2 = 18.18
Before calling maxInFirst: f1 = 9.9 f2 = 23.23
After  calling maxInFirst: f1 = 23.23 f2 = 9.9

Enter Employee Name  : Derek Harter
Enter Employee Salary: 32567.15
After initialization Name: Derek Harter salary: 32567.15
```

NOTE: Now that our programs have more functions than just the `main()` function, the use of the function headers becomes meaningful and required. Make sure that all of your functions have function headers preceding them that document the purpose of the functions, and the input parameters and return value of the function.

Lab Submission

An eCollege dropbox has been created for this lab. You should upload your version of the lab by the end of lab time to the eCollege dropbox named `Lab 11 Pointers and Structs`. Work submitted by the end of lab will be

considered, but after the lab ends you may no longer submit work, so make sure you submit your best effort by the lab end time in order to receive credit.

Requirements and Grading Rubrics

Program Execution, Output and Functional Requirements

1. Your program must compile, run and produce some sort of output to be graded. 0 if not satisfied.

2. 40+ pts. Your `maxInFirst` function must work correctly and take integer pointer parameters as specified.

3. 10+ pts. You should have declared the needed structure correctly as specified.

4. 40+ pts. Your `initializeEmployee` function works, prompts for input from user in the functions as required, and accepts a pointer to an `Employee` struct as its parameters as specified.

5. 10+ pts. You main function demonstrates correctly invoking the functions and displays the required output from doing so.

Program Style

Your programs must conform to the style and formatting guidelines given for this course. The following is a list of the guidelines that are required for the lab to be submitted this week.

1. The file header for the file with your name and program information and the function header for your main function must be present, and filled out correctly.

2. A function header must be present for all functions you define. You must document the purpose, input parameters and return values of all functions. Your function headers must be formatted exactly as shown in the style guidelines for the class.

3. You must indent your code correctly and have no embedded tabs in your source code. (Don't forget about the Visual Studio Format Selection command).

4. You must not have any statements that are hacks in order to keep your terminal from closing when your program exits (e.g. no calls to system()).

5. You must have a single space before and after each binary operator.

6. You must have a single blank line after the end of your declaration of variables at the top of a function, before the first code statement.

7. You must have a single blank space after , and ; operators used as a separator in lists of variables, parameters or other control structures.

8. You must have opening `{` and closing `}` for control statement blocks on their own line, indented correctly for the level of the control statement block.

9. All control statement blocks (if, for, while, etc.) must have `{ }` enclosing them, even when they are not strictly necessary (when there is only 1 statement in the block).

10. You should attempt to use meaningful variable and function names in your program, for program clarity. Of course, when required, you must name functions, parameters and variables as specified in the assignments. Variable and function names must conform to correct `camelCaseNameingConvention` .

11. Put the `*` for pointer variable declarations next to the type declaration, with no space between the type and the `*`. Also please follow the convention of using `Ptr` at the end of names for pointer variables.

Failure to conform to any of these formatting and programming practice guidelines for this lab will result in at least 1/3 of the points (33) for the assignment being removed for each guideline that is not followed (up to 3 before getting a 0 for the assignment). Failure to follow other class/textbook programming guidelines may result in a loss of points, especially for those programming practices given in our Deitel textbook that have been in our required reading so far.