CptS 121 - Program Design and Development

Lab 11: More with Structs

Assigned: Week of November 6, 2023
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. Have a great time! Labs are a vital part to your education in CptS 121 so work diligently.

Tasks:

NOTE: this lab is a prime example for how the lab final could look. Please work through all of the lab and understand it to the best of your ability!

Write a console application that computes and displays the total payroll for hourly employees at a company. Each employee is represented by a record in a file called *payroll.txt*. A record consists of the following:

- Employee name (last, first)
- Employee title (B or M)
- Hours worked
- Rate per hour

You must read in all of the records from the file. These records are representative of a *two* week pay period, where the expected hours worked is ~80 hours. You may assume the file does not consist of more than 200 records. You will need to write an algorithm that traverses through the records and determines payment for each employee based on the following:

Basic Employee (title B):

- Payment = hours worked * rate per hour (excluding overtime)
- Overtime = 1.5 * rate per hour (for each hour over 40 hours worked per week)
- Total payment = payment + overtime

Manager (title M):

- Payment = hours worked * rate per hour (excluding overtime)
- Overtime = 1.8 * rate per hour (for each hour over 40 hours worked per week)
- Total payment = payment + overtime

You will need to write the following information to another file called *paid.txt*:

- Total payroll (sum of all employees' total payments)
- Average total payment
- Max total payment
- Min total payment

Design Requirements:

For this problem you must define the following `struct` type:

```
typedef struct employee
{
	char name[100];		// employee's name - last, first
	char title;		// title 'B' or 'M'
	double hours_worked;	// total number of hours worked
	double payrate;		// pay rate per hour
	double payment;		// total payment for the pay period – you will compute!
} Employee;
```

You must also define an array of `Employees` that contains a maximum of 200 employee profiles. For example,

```
Employee payroll[200]; // the input file may not exceed 200 employee records
```

You must also complete the following:

- Open *payroll.txt* for mode read
- Read all records in *payroll.txt* and store them into the array of `Employees`
- Compute the payment for each employee based on the title, number of hours worked, and pay rate – store the payment back into the `Employee` in the array
- Compute the total or sum of payments for all employees
- Compute the average payment per employee
- Compute the maximum payment
- Compute the minimum payment
- Open *paid.txt* for mode write
- Write the total, average, maximum, and minimum payments to *paid.txt*
- Close *payroll.txt* and *paid.txt*

Sample payroll.txt File:

```
Smith, Susan
B
80.0
17.76
Sanders, Fred
M
87.25
23.45
Kerr, Heidi
M
80.0
47.86
Russo, Rick
B
83.75
12.15
```

Sample paid.txt File (printed to hundredths):

```
Total: $8471.97
Average: $2117.99
Max: $3828.80
Min: $1040.34
```

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, complete two-thirds of the problems, and continue to work on the problems until the TA has dismissed you.