Assignment 3

COMP 2401

Date: October 30, 2017

Due: on November 9, 2017 before 23:55

Submission: Electronic submission on cuLearn.

Objectives:

a. Familiarity with structures and unions.
b. Declaring variables as sub elements using bit fields
c. Formatting output using printf statement
d. Using string functions (strcpy(), strcmp())
e. Pass by reference (an address)
f. Experiment with code reuse

Submission (20 pts)

- Submission must be in cuLearn by the due date.
- Submit a single tar file with all the c and h files.
- Submit a Readme.txt file explaining
    - Purpose of software
    - Who the developer is and the development date
    - How the software is organized (partitioned into files)
    - Instruction on how to compile the program (give an example)
    - Any issues/limitations problem that the user must be aware of
    - Instructions explaining how to use the software

Grading

- 10 points for a readme file
    - 1 point for Purpose of software
    - 1 point for the developer name and development date
    - 2 points for description on how the software is organized (partitioned into files)
    - 2 points for instruction on how to compile the program (give an example)
    - 1 point for any issues/limitations problem that the user must be aware of
    - 2 points for instructions explaining how to use the software

- 5 points for proper submission of files
    - submitting a single tar file and a single readme file
    - using correct file names
- 10 points for using good program layout
    - Adding small functions as needed
    - Easy to read/review code
    - Code documented properly (e.g., purpose of functions, role of variable/parameters, code flow)
    - No "mega" functions
    - No globals

Grading:

You will be graded with respect to what you have accomplished. Namely, you earn points for working code and functionality.

Background

You are tasked to manipulate records of students and employees at Carleton University. The information will be entered by the user using a menu system. As the data is entered your program will accept the data, verify it, where applicable and prompts the user to re-enter it if it is incorrect.

The students and employees share several data fields: First Name, Family Name, and Telephone. Other fields are different: employee records consists of salary, years of service, and level, while student records consists of GPA, Number of Courses taken and tuition fees.

The following information defines the fields:

Common fields

- First Name – 15 characters
- Family Name – 15 characters
- Telephone – 10 characters

Student fields use bit fields to minimize the size

- GPA – range 0-10 (integer)
- Tuition fees – a real non negative number
- Number of courses taken – range 0-40 (integer)

Employee fields use bit fields to minimize the size

- Salary – a real non negative number
- Years of service – range of 0-63 (integer)

- Level – salary level scale range 1-15 (integer)

Tasks

In this assignment you will write a short program to maintain and manipulate the people at Carleton U (students and employees).

Coding Instructions:

1. Comments in Code – as provided in the slides given in class
2. No usage of global variables. All data must be passed or received via function parameters.
3. Write short and simple functions.
4. Using separate files – Create four C (*.c) files and four header files (*.h)
   4.1. uni.c – containing the main program (containing main(argc, argv) and the menu functionality).
   4.2. student files – two files: student.c will contains all the functions for handling student records (e.g., entering student data, printing a student record or searching students by name); student.h will contain the declaration of functions (function prototype) and struct.h
   4.3. employee files – two files employee.c and employee.h which will contain employee related data. Employee.h will contain the file struct.h
   4.4. struct.h – a file containing the declaration of the three structures (student, employee and person)

1) Suggestions –
   a) As you code your small helping functions write small test functions to ensure that the code is correct. This will allow you to focus on the logic of your program without worrying about the simple functions. In particular make sure that you have a few functions that check the validity of the input.
   b) Create function for each menu option. The function should accept as input the array of person and the number of elements in the array.

Tasks

1. **Creating student, employee, person structures (15 pts)**

   Grading (this includes some bonus points)

   - 5 points for correct structure (common fields, using unions, and a discriminant)
   - 2 point for using a define statements for "magic" numbers e.g., length of name
   - 8 pointes – for using proper packing order and bit fields.
     - 3 points for packing the structure in the correct order
     - 5 points for using bit fields correctly (including data types)

   Here you will create the structures to be used by the program. Make sure that you pack the structure as

small as possible (reduce the footprint). Use bit fields to represent small ranges. Organize the structures to reduce memory foot print.

1.1. Create a **student structure** consisting of the student's fields: GPA, number of courses, and tuition fees. Fields names are: gpa, numCourses, tuitionFees respectively.

1.1.1. Structure name is struct student

1.2. Create an **employee structure** consisting of the employee's fields: Salary, years of service, and level. Fields names are: salary, yearsService, and level respectively.

1.2.1. Structure name is struct employee

1.3. Create a **person structure** that consists of the common records: first name, family name and telephone and a union between the student and employee record. Note that you need to ensure that the '\0' character at the end of a string can be stored. Make sure to add a field to discriminate between the employee and student substructures. Discriminating field is emplyeeOrStudent (field is 0 if employee and 1 if student).

1.3.1. Structure name is struct person

1.4. Create an array that can hold 20 records of person. The array name is person

2. Populate the array

Populate the records in the array Using the function populateArray() in file populate_array.c (file populate_array.h contains that function prototype). Note that you will have to link the file with your code.

3. Menu (5 pts)

Create a menu function that will allow the user to manipulate the lists.

3.1. Create a menu function (int menu()) which will return the menu option that the user has selected.

3.2. Display the menu options:

3.2.1. 1. Print all employees

3.2.2. 2 Print all students

3.2.3. 3 Search students using Family Name

3.2.4. 4 Summary of Data

3.2.5. 0. Quit

3.3. The program will display the menu to the user and ask the user to enter an option. If the selected option is valid that the system will execute the option. Otherwise, the system will prompt the user that the option is not valid and then redisplay the menu.

4. Menu actions (55pts)

4.1. (**5 pts**) Create a switch statement in the main function, which will process the menu item selected by the user. Each "case" will invoke a corresponding function that will execute one of the actions described in 3.2 – 3.6.

Here you will be responding to the action selected by the user as follows. Allow the user to enter up to 20 records.

4.2. Print students list (5pts)

Print all the students. Hints – a. create a function to print a student; b. use sprintf to print the first name and the last name to a temporary string. Then using the temporary string print the name using the formatting.

Print format

```
First Name Family Name (33 char), GPA:xxx, Courses:xxx, Tuition: xxxxx.xx
    For example:
John Dilbert                    GPA:  7, Courses: 13, Tuition: 3450.34
Jane Smith                      GPA: 10, Courses: 15, Tuition: 3450.00
```

Grading:

- 5 points for printing as required

4.3. Print employee list (5pts)

Print all the employees. Hints – a. create a function to print an employee; b. use sprintf to print the first name and the last name to a temporary string. Then using the temporary string print the name using the formatting.

Print format

```
First Name Family Name (33) characters, Years:xxx, Level:xxx, Salary:xxxxxx.xx
For example
John Dilbert                     Years: 58, Level: 13, Salary:   450.34
Jane Smith                       Years: 47, Level: 10, Salary:133450.00
```

Grading:

- 5 points for printing as required

4.4. Finds student by Family Name (10pts)

4.4.1. Prompt the user to enter a family name

4.4.2. Search the list for all students with a matching family name and print their record as above

Grading:

- 5 points for finding and printing a matched student
- 5 points for using points arithmetic to traverse the array

4.5. **Summary of Records (25 pts)** – here you will provide a summary of all the information that is available

Output:

Total number of records: xx

Students Stats:

Number of students: xx

Average GPA: xx.xx, Average Number of courses: xx.xx, Average Tuition Fees: xxxx.xx

Employees Stats:

Number of employees: xx Min Level: xx Max Level: xx

Average Years of Service: xx.xx, Average Salary: xxxxx.xx

Grading:

- 2 points for correctly printing total number of records
- 2 points for formatting
- 10 points for student summary
    - 1 points for correctly printing total number of students
    - 3 points for correctly printing average GPA
    - 3 points for correctly printing average number of courses
    - 3 points for correctly printing average tuition fees
- 11 points for employee summary
    - 1 points for correctly printing total number of employees
    - 2 points for correctly printing min level
    - 2 points for correctly printing max level
    - 3 points for correctly printing average Years of Service
    - 3 points for correctly printing average salary

4.6. Quit (5 pts)

4.6.1. Warn the user that he/she is about to quit and ask whether to proceed (y/n)

Grading:

- 5 points for correctly warning the user and quitting