CPE 100 Introduction to Computer Programming
International Sections August 2020
Laboratory Exercise 10

Objective

This lab gives you practice using structures, and reading and writing text files. It also uses command line arguments. There are two pages to these instructions.

Instructions

Summary: Write a program call **daranang.c** that will read data about movie stars from a text file into an array of structures that you define in your program. After reading all the information in the file, your program will write a new file that prints the information about each movie star in a more convenient and useful format.

Details:

1. Define a structure to hold information about a movie star. The information to be stored includes *first name*, *last name*, *gender* (M or F), *number of movies made*, and *popularity rating* (1 to 5 stars). For example, your structure definition might look like this:

```
typedef struct
{
     char firstname[32];
     char lastname[32];
     char gender;
     int movieCount;
     int rating;
} MOVIESTAR_T;
```

2. Declare an array of **MOVIESTAR_T** structures that can hold information concerning **10** movie stars.

3. Download the two test data files (**moviestar1.in** and **moviestar2.in**) from the course web site. Each line in these files contains information for one movie star. For example:

```
Angelina Jolie F 22 4
```

The fields on the line represent the first name, last name, gender, number of movies and rating.

4. Use command line arguments to allow the user to specify the input and output filenames. The user will run the program as follows:

```
./daranang moviestar1.in moviestar1.out
```

You can tell whether the user included the correct arguments by checking the value of **`argc.`** If **`argc`** is less than 3, give an error message telling the user she must type an input filename and output file name, then **exit the program**.

5. If the arguments are supplied correctly, try to open the input file that the user specified. Check if the open operation was successful. If an error occurred, print a message and exit.

6. In a loop, read each line of the file, one by one, and parse the different items of information (first name, last name, etc.). To do this, use **fgets** and then **sscanf,** not **fscanf**. Store this information in the different elements of the current **MOVIESTAR_T** structure in the array. The loop should continue until one of two things happens:

a. You have read 10 lines (the maximum that will fit in the array), ***or***
b. You have reached the end of the file (when you do, the **fgets** function will return NULL).

Note that if a file contains more than 10 movie stars, you will only process the first 10.

After exiting the loop, close the input file.

7. Now try to open the output file. Check for success. If the open operation is not successful, print an error message and exit.

8. Enter a loop in which you print the data from each structure that you read into the output file, in a nice form. (Hint: this will be a **for** loop since you will know how many stars you read.) For instance, the output for one structure might look like this:

```
Movie star #1
Name: Angelina Jolie
Gender: F
Number of movies: 22
Popularity rating: 4
```

You can decide what format to use for the output, but you should definitely have more than one line of text per movie star.

9. When you have finished writing the data to the output file, close the output file and exit the program.

10. Run your program on both **moviestar1.in** and **moviestar2.in**. Be sure the results are correct.

Suggestion: First write and test the code for reading the input file. When you're sure that is working, write the code for creating the output file.

For more of a challenge:

Modify your program to use functions to parse and to print the structure information. For instance:

```
/* Initialize one movie star in the array ('which') based on line read
 * from the file ('inputline')*/
parseStar(char inputline[], MOVIESTAR_T stars[], int which);

/* Print information for one movie star in the array ('which') into the file
* which is already open with file handle 'pOutput' */
printStar(FILE* pOutput, MOVIESTAR_T stars[], int which);
```

How to submit your work

You should submit three files: **daranang.c** (your source code), **moviestar1.out** and **moviestar2.out** (that is, the results of running your program on *moviestar1.in* and). Use the usual link on the web page to submit your work.