CIS 236 – Programming in C

Program:	**Week 13**
Points:	**20**
Chapters:	**11**

Description

Write a program that stores information about movies, using an array of structures and a text file.

This program modifies the Week 12 program. If you have not finished the Week 12 program, go back and finish it first.

You must use functions as described in the requirements below.

Extra credit is available for this assignment.

Learning Objectives

In this assignment, you will:

- Use an array of programmer-defined structures
- Use a repetition control structure
- Use a function using an output argument to get data from the user
- Use a function with an array argument to display output to the screen
- Read from a text file using fscanf
- Read from a text file without using fscanf (extra credit)
- Write to a text file (extra credit)
- Use enumerated data types (extra credit)

Requirements

Instead of getting the movie data from the user, read the data from a plain text file.

Your code **must** use these functions, using these names (in addition to main):

1. **getMovieInfo** (A different version than the week 12 homework)
2. **displayMovieInfo** (Same function as the week 12 homework. It may need to change if you do Extra Credit 2 or 3).

The structure requirements are the same as the week 12 homework:

The information for each movie must be stored in a structure containing the title of the movie (a string of size 21) and the number of downloads. The information for all movies is stored as an array of these structures. Be sure to use the appropriate data type for the number of downloads.

Note that main uses a slightly different set of requirements than the week 12 homework, so be sure to read the information in that section below!

Requirements for the getMovieInfo Function

Purpose:	This function **opens** and **reads** a plain text containing movie information and stores the information in the appropriate position in the array. Then it **closes** the file.
Parameters:	1. array of movie structures 2. size of the array
Algorithm:	Open the file appropriately. Be sure to verify that the file opened successfully. Then, **within a loop that reads until the end of the file**, use **fscanf** to read the movie titles and numbers of downloads from a plain text file with the following format: TitleOfMovie NumberOfDownloads TitleOfMovie NumberOfDownloads TitleOfMovie NumberOfDownloads TitleOfMovie NumberOfDownloads Note that the title of the movie will NOT have any spaces in it. Always set the last element in the string to the null character, regardless of the length, although no message is required. All information should be stored in the output parameter. **Continue to read the movie information until the end of the file (EOF). DO not use a counter-controlled loop for this!** Then close the file.
Return value:	None

Requirements for main:

1. Create an array of movie structures of size 10. You can assume that there will not be more than 10 movies listed in the input file.
2. Call the getMovieInfo function and pass the array of movie structures.
3. Call the displayMovieInfo function.
4. Extra credit 1 – call the writeToFile function.

Extra Credit 1 – Max 5 Points – Write the Movie Information to an Output File

Purpose:	Use a function called **writeToFile** to create a text file and write the contents of the movie array to this file. Then close the file. DO NOT USE THE SAME FILE NAME AS YOUR INPUT FILE.
Parameters:	1. array of movie structures 2. size of the array
Algorithm:	Use a counter-controlled loop to write the contents of the array to a plain text file, using the same file format as shown in the requirements for the getMovieInfo function. Be sure to only print existing data. Do not print empty structures. Then close the file.
Return value:	None

Extra Credit 2 – Max 10 Points – Add Genre to the Movie Information

Add information about genre to the movie information data.

The genre is stored as an enumerated type in the structure, using these enumeration constants, in this order: action, animation, drama, sci-fi, superhero.

In this version of the input file, the **genre is stored as a number** after the number of downloads:

TitleOfMovie NumberOfDownloads Genre
TitleOfMovie NumberOfDownloads Genre
TitleOfMovie NumberOfDownloads Genre
TitleOfMovie NumberOfDownloads Genre

Update the getMovieInfo function to handle this additional item. You can assume the genre value will be a valid genre.

The displayMovieInfo must use a **switch statement** to determine which **genre text** to print. Remember, the genre for each movie is stored as an enum value, not as a string. Your output should display actual words, like Action or Drama.

Extra Credit 3 – Max 15 Points – Alternate Formatting of Input File

Adjust the getMovieInfo function to handle a file with the following format:

Title of Movie,NumberOfDownloads
Title of Movie,NumberOfDownloads
Title of Movie,NumberOfDownloads
Title of Movie,NumberOfDownloads

Note that the title of the movie **may have spaces** in it. The title and the number of downloads are separated by only a comma.

Sample Run – Required

```
Title                     Downloads
=====                     =========
TheSuicideSquad            10000000
BirdsofPrey:Harle           3000000
Avengers:Endgame            3000000
WonderWoman1984               17340
ToyStory4                  10000000
```

Sample Run – Extra Credit 2

```
Title                     Downloads              Genre
=====                     =========              =====
TheSuicideSquad            10000000              Action
BirdsofPrey:Harle           3000000              Superhero
Avengers:Endgame            3000000              Sci-Fi
WonderWoman1984               17340              Superhero
ToyStory4                  10000000              Animation
```

Sample Run – Extra Credit 3

```
Title                     Downloads
=====                     =========
The Suicide Squad          10000000
Birds of Prey: Harle        3000000
Avengers: Endgame           3000000
Wonder Woman 1984             17340
Toy Story 4                10000000
```

Requirements for Full Credit on This Project

SUBMIT YOUR OWN WORK – Plagiarism is not tolerated in this course. Please review the section on the Academic Honor Code in the syllabus. I will not hesitate to drop you from this class if you submit a program that is plagiarized.

COMPLETE AND ACCURATE – Your program must compile, execute, and give accurate output.

FOLLOW ALL REQUIREMENTS ACCORDING TO THE INSTRUCTIONS – Follow the instructions as written for completing this project, even if you [think you] know a "better" way to do something.

COMMENTS – Include comments in your code. There must be a comment at the top of your program that includes your name, the program number, and a description of the program. There must be comments at each important step in your program that describes that step. Every variable must include a comment describing its purpose.

BEST PRACTICES – Follow best practices in C programming as discussed in class and in the textbook, including, but not limited to, appropriate use of white space, indenting, alignment, meaningful identifier names, etc. Points will be deducted for sloppy code that is hard to read, even if it works, so pay attention to these details.

SUBMIT ONLY .c SOURCE CODE – Pay attention to the file extension of the source code file you submit. I will deduct points for not following this requirement.

SUBMIT ALL FILES BEFORE THE DUE DATE – Submit your .c source code file to the dropbox for this assignment on Canvas before the due date. Do not submit executable files. Do not submit project files from an IDE. I will not accept links to online storage.