Programming Assignment 1

C++ Basics, Classes, and Aggregation

Due 9//2024 @ 11:59 PM

Late/Redo deadline 10/24/2023 @ 11:59 PM

Assignment Objectives

- Provide practice with C++ strings, IO, command line arguments, and other C++ basic syntax.
- Provide practice with makefiles and multifile programs.
- Provide practice with class building skills.
- Provide practice with class aggregation skills.
- Provide practice with reading technical documentation.

Important Notes

- The assignment contains a series of files. One file is an example executable called genresExampleExe. You can run it by doing ***./***genresExampleExe. Also provided is a test python file called PA1_endToEnd.py. It can be run by typing python3 PA1_endToEnd.py in the terminal and it must be in the same directory as the files you want to test. Each test will give you a test name, which you can use to go back and fix your code.
    - If you get a permission denied error, run **chmod u+x <filename>** where <filename> is PA1_endToEnd.py. (<> angled brackets indicate to replace the entire entry with the requested information, including the brackets)
    - The example executable **will not work on mac**. You will need to run it in ubuntu with your code.
- 50% of your grade will be for passing the unit tests and a longer end to end test. The other half will be for program structure, which we will grade using the rubric posted on WebCampus with the assignment. Make sure you are following the styling conventions laid out on the c++ styling conventions page on WebCampus.
- I have provided a csv file named IMDB-Movie-Data.csv that contains the data for your assignment. **Do not hardcode its name in.** The name should be provided as a command line argument, and the end to end test will check your code using a file with a different name *with the exact same columns*.
- You have been asked to create an array of type Movie. Do not use strings in place of a Movie object.
- Code that does not compile and run until the main menu will receive a 0.
- Please name your executable **genres**

Program

Your program should contain the following files:

- main.cpp
- movie.h
- movie.cpp
- genre.h
- genre.cpp
- helpers.h
- helpers.cpp
- makefile

Please **do not** include your PA1_endToEnd.py, executable or the csv file. You are welcome to add additional files where it makes sense to do so, but these files are the bare minimum expectation.

Programming Problem

Write a program that reads a csv file and then prompts the user to select a movie genre to view. Here's example output, though you are welcome to format it in whatever way you see fit:

1. **SciFi**
2. **Fantasy**
3. **Biography/Documentary**
4. **Romance**
5. **Comedy**
6. **Horror**
7. **Drama**
8. **Family**
9. **Exit**

==========
Which genre of movie would you like to view?

The program should then allow the user to enter a value, 1-9, to view **on the same line as the prompt to view at the end of the display**. If a user makes an invalid entry, the program should repeatedly display the menu until a valid option is chosen. If the user chooses to exit, the program should gracefully end- do not use the exit keyword. Otherwise, the user should be prompted to choose a method of displaying the movies. Here's example output, though you are welcome to format it in whatever way you see fit:

1. **Alphabetically by title**
2. **Alphabetically by director last name**

3. **Numerically by year**
4. **Numerically by rating**
5. **Numerically by metascore**
6. **Exit**

==========

How would you like to view <the genre chosen by user> movies?

The program should then allow the user to enter a value, 1-6, to view on the same line as the prompt to view at the end of the display. If a user makes an invalid entry, the program should repeatedly display the menu until a valid option is chosen. If the user chooses to exit, the program should gracefully end- do not use the exit keyword. Otherwise, the program should display the titles of all movies of that genre sorted using the method chosen by the user. Below, you will find 5 examples (one for each sort). These examples are not all inclusive, they're just meant to show you how the order would change. The information displayed should match the examples, but the format is your choice.

We'll use the following movies for our comparison:

Rank	Title	Genre	Description	Director	Actors	Year	Runtime	Rate	Votes	Rev (million)	Meta score
1	Guardians of the Galaxy	Action, Adventure,Sci-Fi	A group of intergalactic criminals are forced to work together to stop a fanatical warrior from taking control of the universe.	James Gunn	Chris Pratt, Vin Diesel, Bradley Cooper, Zoe Saldana	2014	121	8.1	757074	333.13	76
13	Rogue One	Action, Adventure,Sci-Fi	The Rebel Alliance makes a risky move to steal the plans for the Death Star, setting up the epic saga to follow.	Gareth Edwards	Felicity Jones, Diego Luna, Alan Tudyk, Donnie Yen	2016	133	7.9	323118	532.17	65
25	Independence Day: Resurgence	Action, Adventure,Sci-Fi	Two decades after the first Independence Day invasion, Earth is faced with a new extra-Solar threat. But will mankind's new space defenses be enough?	Roland Emmerich	Liam Hemsworth, Jeff Goldblum, Bill Pullman ,Maika Monroe	2016	120	5.3	127553	103.14	32
33	X-Men: Apocalypse	Action, Adventure,Sci-Fi	After the re-emergence of the world's first mutant, world-destroy	Bryan Singer	James McAvoy , Michael Fassbe	2016	144	7.1	275510	155.33	52

			er Apocalypse, the X-Men must unite to defeat his extinction level plan.		nder, Jennifer Lawrence, Nicholas Hoult						
36	Captain America: Civil War	Action, Adventure,Sci-Fi	Political interference in the Avengers' activities causes a rift between former allies Captain America and Iron Man.	Anthony Russo	Chris Evans, Robert Downey Jr.,Scarlett Johansson, Sebastian Stan	2016	147	7.9	411656	408.08	75

Example 1- alphabetically by title

1. **Captain America: Civil War**
2. **Guardians of the Galaxy**
3. **Independence Day: Resurgence**
4. **Rogue One**
5. **X-Men: Apocalypse**
6. **Exit**

==========

Which movie would you like to view?

Example 2- numerically by year (lowest first)

1. **Guardians of the Galaxy**
2. **Captain America: Civil War**
3. **Independence Day: Resurgence**
4. **Rogue One**
5. **X-Men: Apocalypse**
6. **Exit**

==========

Which movie would you like to view?

Example 3- numerically by rating (highest first)

1. **Guardians of the Galaxy**
2. **Captain America: Civil War**
3. **Rogue One**

4. **X-Men: Apocalypse**
5. **Independence Day: Resurgence**
6. **Exit**

==========
Which movie would you like to view?

For numerical sorts, tie break using titles in alphabetical order.

The program should then allow the user to enter a value, 1 to n+1 (n = number of movies of that genre), to view on the same line as the prompt to view at the end of the display. If a user makes an invalid entry, the program should repeatedly display the menu until a valid option is chosen. If the user chooses to exit by entering n+1, the program should gracefully end- do not use the exit keyword. Otherwise, the program should display the movie title, director, actors, year, rating, meta score, and description for the selected movie followed by a prompt to select another movie. **The formatting of this display must follow the exact formatting you see below.**

Guardians of the Galaxy
Director: James Gun
Actors: Chris Pratt * Vin Diesel * Bradley Cooper * Zoe Saldana
Year: 2014 Rating: 8.1 Meta Score: 76
Description: A group of intergalactic criminals are forced to work together to stop a fanatical warrior from taking control of the universe.

Would you like to select another movie(y/n)?

This display has been implemented for you in the movie class. DO NOT MODIFY this display out method.

The program should then allow the user to enter y or n. If a user makes an invalid entry, the program should repeatedly display the "would you like to select" prompt until a valid entry is made. If the user chooses n, the program should gracefully end- do not use the exit keyword. Otherwise, the program should loop again starting from the genre selection menu (which you can consider to be the main menu).

Program Requirements

- The genre and sort menus should use the same numbering for options.
- The file containing the movie data should be read outside of the main program loop (ie, read only once)

- Your program should have two classes: Movie and Genre; property names are given in bold
    - Movie consists of a **title**, **genres** array, **actors** array, **description**, **director**, **year**, **rating**, **metascore**, **actorCount,** and **genreCount**
    - Genre consists of **genreOfObject**, **moviesOfGenre**, and **numMoviesInGenre**
- There should be a single Genre object created after the user chooses a genre to view that contains the name of the genre and the array of Movies of that genre type.
- The Movie objects should **only** contain the properties necessary for the program to function, and the properties should NOT all be strings (see below on getline and casting).
- You should use the string library when sorting. The string library allows you to use normal operators (such as <, >, ==) directly on two strings. You can also use tolower to lowercase the strings (temporarily) for direct comparison. Remember, A is not the same as a, so you want everything to be formatted the same before comparing. You may include the cctype library to use tolower.
- Reading a csv file in c++ is done the same way that it's done with a txt file. csv stands for comma separated values, so every entry in the csv is separated by commas so that when the file is opened in excel (or other spreadsheet software), the columns are automatically created. You can include the fstream library to read the file.
    - You can open the file as a text file by right clicking on the file and selecting open with, then the text editor- this will allow you to see the raw file with all of the commas between entries.
    - There are several columns with multiple string entries that are separated by commas. To make sure they all end up in the same column entry, the entire entry is surrounded by quotation marks.
    - That means that each entry in the csv follows this format:

1,Guardians of the Galaxy,"Action,Adventure,Sci-Fi",A group of intergalactic criminals are forced to work together to stop a fanatical warrior from taking control of the universe.,James Gunn,"Chris Pratt, Vin Diesel, Bradley Cooper, Zoe Saldana",2014,121,8.1,757074,333.13,76

    - You will need to be smart about how you parse the data that you read from the file. The easiest option is to use getline because you can choose a character to split the line of info on. You will want to cast ints back into ints using stoi and cast floats/doubles into floats using stof.
    - Columns that have multiple entries (such as genre or actor) should be treated as separate strings that are added to an array. So, for example, an array for genre would look like this: ["Action", "Adventure", "Sci-Fi"]

- You will need to sort Movie objects based on the sorting option chosen by the user. You will use bubble sort**.** Bubble sort is a pretty common algorithm to sort data. You can find a great description here- you basically use two for loops to compare one element in the array to each other element in the array. In this example, they are directly comparing entire variables. You will not be able to do that- you should be using the getters to get the property that the user chose to sort on, and compare those values.

- You must include the following helper functions, they must match the specifications given, and your parameters should be provided in the order shown below.

Method: bubbleSortSelection
Return Type: None
Parameters: userSortChoice (int), genreObject (Genre&)
Pre-Condition: An unsorted array of Movies has been created from the csv file and stored in a Genre object after the user chose a genre.
Post-Condition: Array of Movies sorted within the genre object
Description: Sorts the array of Movies by using the bubble sort algorithm; the sort type is used to determine what to compare when sorting.
Methods Called: Your choice.

Method: readCSV
Return Type: int
Parameters: movieArray(Movie*), inputFile (ifstream&)
Pre-Condition: movies array has been initialized in main, input stream was opened in main using command line argument
Post-Condition: movie array is filled with the movies from the csv file
Description: uses the opened inputFile stream to read the data from each line of the file into a Movie object that is stored in the passed movie array; only uses the data from the file necessary to program operation. Returns number of movies read from the file.
Methods Called: Your choice.

Method: constructGenre
Return Type: Genre
Parameters: movieArray(Movie*), numMovies(int), genre(string)
Pre-Condition: movies array has been filled and user has selected genre
Post-Condition:a genre object containing only movies of the user selected genre is created
Description: iterates through every movie in movieArray and checks all genres for that movie. If the movie has the user selected genre as one of its genres, then it should be added to a genre object.
Methods Called: Your choice.

The PA1_endToEnd.py tests for a very specific series of inputs. The inputs are provided here so that you may test whether your output looks the same as the provided example executable or not. The way to read the inputs is the number refers to the number selection in the menu. ‘y’ or ‘n’ refers to choosing to select another movie or not. Also be aware these inputs are being run against a different file so they may not correspond one to one to your own code.

Test 1 (alphabetical sort by title):
7 1 3 n

Test 2 (numerical sort by year):
7 3 1 n

Test 3 (numerical sort by rating):
7 3 2 n

Test 4 (exit from main menu):
9
Test 5 (exit from sort menu):
7 6

Test 6 (exit from movie selection menu):
7 5 5