INTEC

EDUCATION COLLEGE

USCS22
GROUP ASSIGNMENT 1

PROGRAMME:
ADTP

CLASS:
NEVADA

NAME:
MUHAMMAD AIMAN FIRDAUS BIN KHOIRI (23032631)
MUHAMAD IRFAN BIN MUHAMAD NASIR (23038820)
KHILFI BIN KHAIRUL AMIN (230339065)

LECTURER'S NAME:
MS. HARYATI BINTI MD KASIM

DATE OF SUBMISSION:
4 NOVEMBER 2024

TABLE OF CONTENTS

INTRODUCTION:

This project is a comprehensive movie management system designed to simplify the organization and tracking of movies in a collection. It enables users to add, update, and delete movies, as well as manage associated information like genres, release dates, ratings, and more. The system allows for efficient searching and sorting of movies, providing a user-friendly interface that caters to both casual users and film enthusiasts. The goal of this project is to streamline the management of movie data, making it easier to browse and maintain a movie library.

Problem Statement:

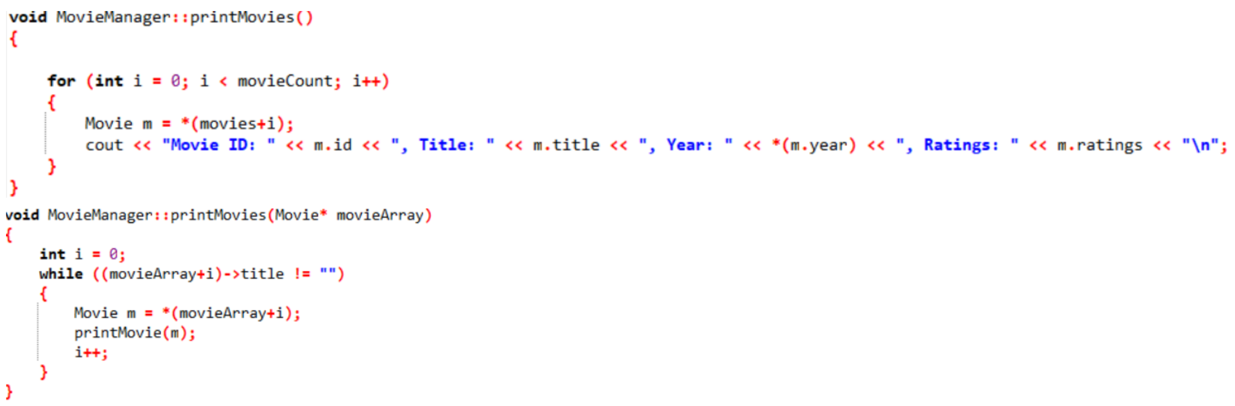
Developing a system related to movies; by using C++ and incorporating key concepts such as string input and output, structures, classes, arrays, pointers, recursion, and file handling. The program will allow users to manage a list of movies, including some functions:

CODE EXPLANATION:

Functions

1. **printMovies()**

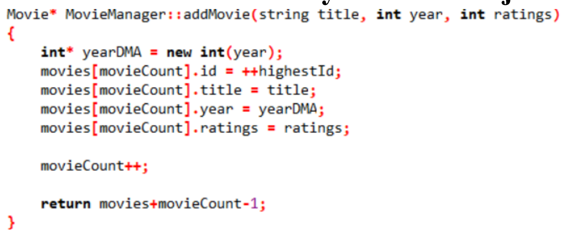
This function will display all the movie details such as title, year, id, ratings stored in movies array. There is an overload function which accepts specific arrays searched by movie title, year, id, or ratings.



```cpp
void MovieManager::printMovies()
{

    for (int i = 0; i < movieCount; i++)
    {
        Movie m = *(movies+i);
        cout << "Movie ID: " << m.id << ", Title: " << m.title << ", Year: " << *(m.year) << ", Ratings: " << m.ratings << "\n";
    }
}
void MovieManager::printMovies(Movie* movieArray)
{
    int i = 0;
    while ((movieArray+i)->title != "")
    {
        Movie m = *(movieArray+i);
        printMovie(m);
        i++;
    }
}
```

2. **addMovie()**

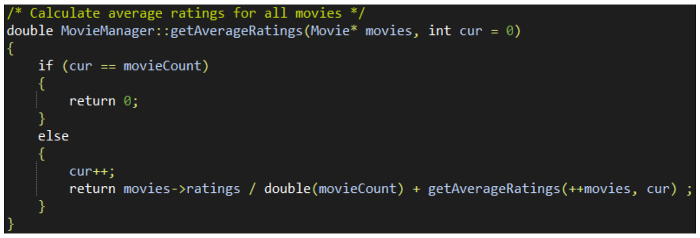
This function accepts input from users such as title,year,ratings to be saved in the stated index of the array in struct object which is movies.



```
Movie* MovieManager::addMovie(string title, int year, int ratings)
{
    int* yearDMA = new int(year);
    movies[movieCount].id = ++highestId;
    movies[movieCount].title = title;
    movies[movieCount].year = yearDMA;
    movies[movieCount].ratings = ratings;

    movieCount++;

    return movies+movieCount-1;
}
```

3. **Calculate average rating function**

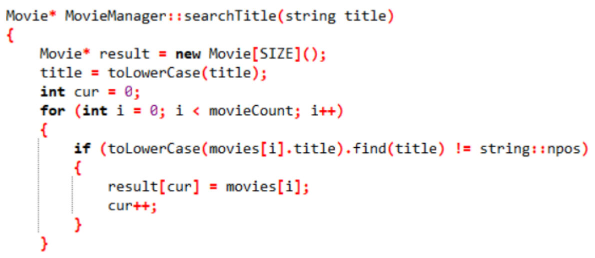
The average rating function will use a recursive function. The function will take the movies pointer and current index as input. Then, the function calculates average ratings by dividing total ratings by number of movies.



```
/* Calculate average ratings for all movies */
double MovieManager::getAverageRatings(Movie* movies, int cur = 0)
{
    if (cur == movieCount)
    {
        return 0;
    }
    else
    {
        cur++;
        return movies->ratings / double(movieCount) + getAverageRatings(++movies, cur) ;
    }
}
```

4. **Search functions**

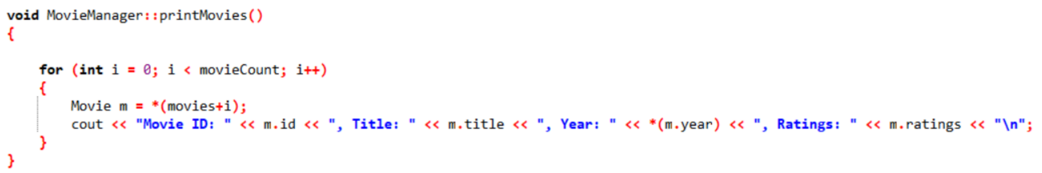
There are some search functions such as searchId, searchTitle, searchYear, and searchRatings which allow users to search movies by its id, title, released year and ratings. Search made using title uses built in function npos and also find. Function find attempts to find substring titles within movies[i].title. If the substring matched so its index will be assigned to the array list named result. Otherwise, the search will continue until the last index in the movies array.



```
Movie* MovieManager::searchTitle(string title)
{
    Movie* result = new Movie[SIZE]();
    title = toLowerCase(title);
    int cur = 0;
    for (int i = 0; i < movieCount; i++)
    {
        if (toLowerCase(movies[i].title).find(title) != string::npos)
        {
            result[cur] = movies[i];
            cur++;
        }
    }
```

5. **save()**

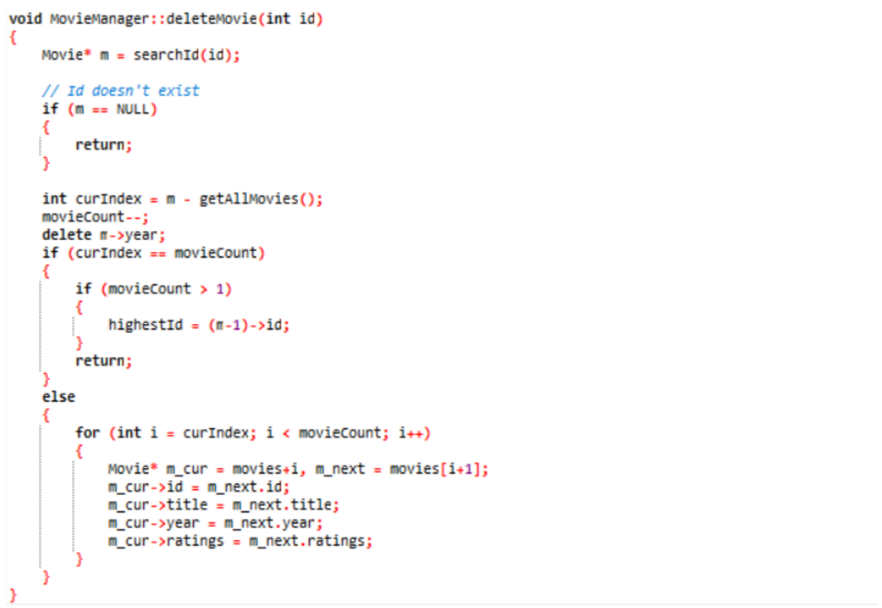
save() writes all the movie details such as title, year, id, ratings in movies array to “movies.txt”



```
void MovieManager::printMovies()
{

    for (int i = 0; i < movieCount; i++)
    {
        Movie m = *(movies+i);
        cout << "Movie ID: " << m.id << ", Title: " << m.title << ", Year: " << *(m.year) << ", Ratings: " << m.ratings << "\n";
    }
}
```

6. **delete()**

   delete() removes the selected movie details based on the requested id.

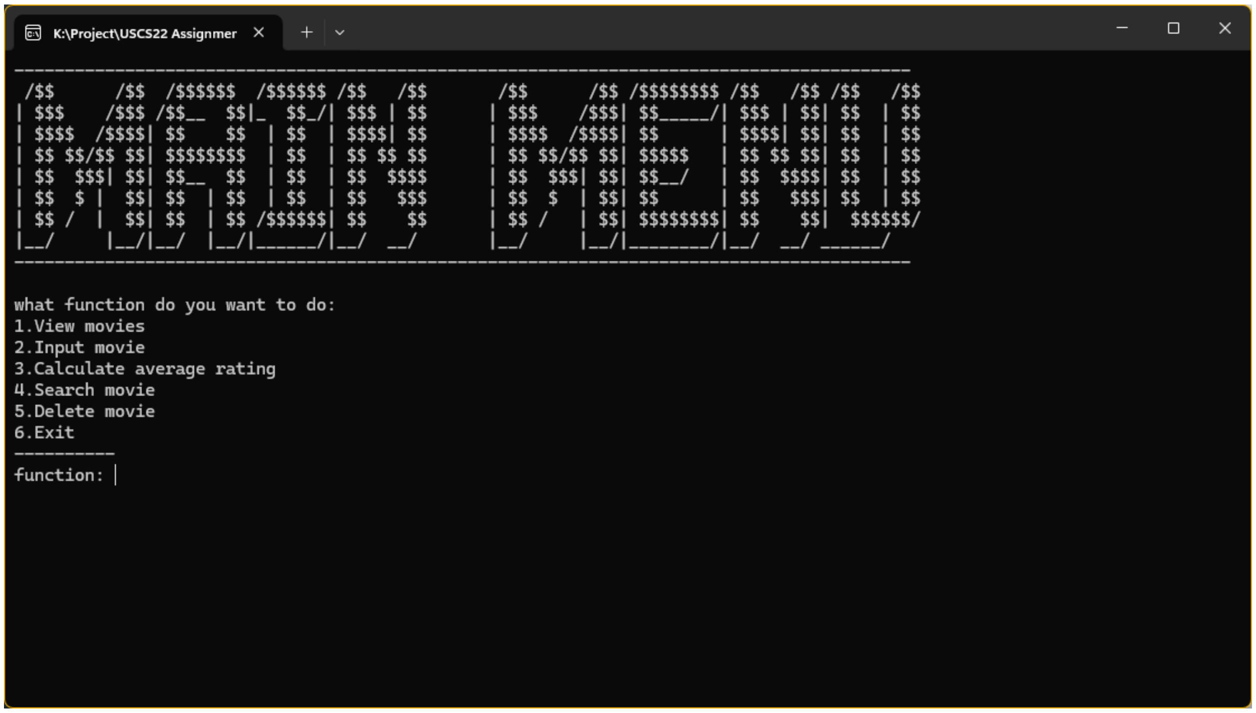


```
void MovieManager::deleteMovie(int id)
{
    Movie* m = searchId(id);

    // Id doesn't exist
    if (m == NULL)
    {
        return;
    }

    int curIndex = m - getAllMovies();
    movieCount--;
    delete m->year;
    if (curIndex == movieCount)
    {
        if (movieCount > 1)
        {
            highestId = (m-1)->id;
        }
        return;
    }
    else
    {
        for (int i = curIndex; i < movieCount; i++)
        {
            Movie* m_cur = movies+i, m_next = movies[i+1];
            m_cur->id = m_next.id;
            m_cur->title = m_next.title;
            m_cur->year = m_next.year;
            m_cur->ratings = m_next.ratings;
        }
    }
}
```

7. **Exit function**

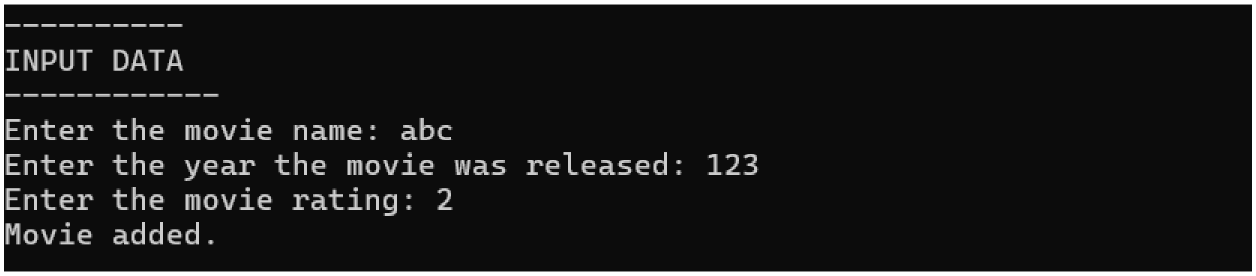
   Exit function will exit the while loop imposed that keeps the main menu repeating.

OUTPUT EXAMPLE:



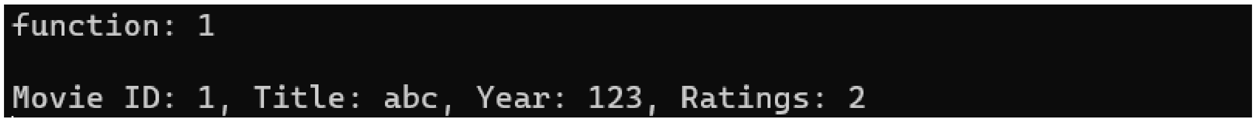
```
K:\Project\USCS22 Assignmer
-----------------------------------------------------------------------------------------
 /$$      /$$  /$$$$$$  /$$$$$$ /$$   /$$       /$$      /$$ /$$$$$$$$ /$$   /$$ /$$   /$$
| $$$    /$$$ /$$__  $$|_  $$_/| $$$ | $$      | $$$    /$$$| $$_____/| $$$ | $$| $$  | $$
| $$$$  /$$$$| $$   $$  | $$  | $$$$| $$      | $$$$  /$$$$| $$      | $$$$| $$| $$  | $$
| $$ $$/$$ $$| $$$$$$$$  | $$  | $$ $$ $$      | $$ $$/$$ $$| $$$$$   | $$ $$ $$| $$  | $$
| $$  $$$| $$| $$__  $$  | $$  | $$  $$$$      | $$  $$$| $$| $$__/   | $$  $$$$| $$  | $$
| $$  $ |  $$| $$  | $$  | $$  | $$   $$$      | $$  $  | $$| $$      | $$   $$$| $$  | $$
| $$ /  |  $$| $$  | $$ /$$$$$$| $$    $$      | $$ /   | $$| $$$$$$$$| $$    $$|  $$$$$$/
|__/     |__/|__/  |__/|______/|__/  __/      |__/     |__/|________/|__/  __/ ______/
-----------------------------------------------------------------------------------------

what function do you want to do:
1.View movies
2.Input movie
3.Calculate average rating
4.Search movie
5.Delete movie
6.Exit
----------
function: |
```

First of all, user will be asked to choose operations based on the given option



```
----------
INPUT DATA
------------
Enter the movie name: abc
Enter the year the movie was released: 123
Enter the movie rating: 2
Movie added.
```

Next, user is requested to provide movie details and they will be saved to a text file (“movies.txt”)



```
function: 1

Movie ID: 1, Title: abc, Year: 123, Ratings: 2
```

All the movie details are displayed.