تمرین 1 _ برنامه نویسی پیشرفته

موضوع تمرین : آغاز شیء گرایی

مدرس

دکتر مرتضی یوسف صنعتی

دانشجو

سیدعلی زین الدینی _ 9912358021

بهار 1400

مقدمه

شی گرایی از مباحث مهم در زمینه برنامه نویسی است. دید شیءگرا می تواند از مهم ترین و تاثیر گذار ترین مباحث در زمینه برنامه نویسی برای یک توسعه دهنده باشد.

این تمرین به گونه ای طراحی شده است که همه نوشتار و سینتکس زبان C++ را در بر بگیرد و در ترکیب آن با مباحث شیءگرایی یک برنامه هدفمند تولید شود.

این دید شیء گرا نه فقط در زبان برنامه نویسی گسترده C++ استفاده می شود بلکه یک مبحث مشترک بین اکثر زبان های برنامه نویسی سطح بالا است.

درک عمیق از شیء گرایی می تواند مسائل را برای ما تفکیک کند و باعث می شود بتوانیم مسائل را به شدت به دنیای حقیقی نزدیک کنیم و آن ها را حل کنیم.

در این مبحث برنامه نویس می تواند یک آفریننده واقعی باشد و جهانی مانند دنیای حقیقی خلق کند.

با توجه به ماهیت این تمرین، با توسعه این برنامه توسط یک زبان سمت سرور مانند php یا فریمورک ها و کتابخانه های تحت وب C++ و ایجاد تغییرات مناسب در سطح برنامه و اتصال به دیتابیس می توان یک وب سایت شبیه به imdb برای رتبه دهی به فیلم ها و سریال ها خلق کرد.

شرح پروژه

این برنامه سعی شده است که استاندارد های کدنویسی تمیز از همان ابتدا رعایت شود. برنامه شامل 29 تابع است. سعی شده کار های برنامه به قطعات کوچک تر شکسته شود و هر تابع فقط یک کار انجام دهد.

16 تابع متعلق به کلاس Movie است که شامل توابع set و get هستند که در فایل هدر movie.hpp الگوی اولیه (prototype) این توابع نوشته شده است.

MOVIE.HPP

```
#include<iostream>
#include<cctype>
#include<stdexcept>
#include<random>
#include<cstring>
#include<ctime>
#include<iomanip>

#ifndef MOVIE_H
#define MOVIE_H

class Movie
    {
    public:
        //set data
        // Movie(std::string tempname, unsigned int tempscore, unsigned int tempy
ear, unsigned int temptime, std::string directortemp, std::string countrytemp); /
/ constructor
        void set_movie_name(std::string);
        void set_score(float);
        void set_year(unsigned int);
        void set_time(unsigned int);
        void set_director(std::string);
        void set_country(std::string);
        void set_random_id(); // assign an random id for any movie
        void set_score_details();
        // end set

        //get data
```

```
        std::string get_movie_name() const;
        float get_score() const;
        unsigned int get_year() const;
        unsigned int get_time() const;
        std::string get_director() const;
        std::string get_country() const;
        unsigned int get_id() const;
        void get_score_details();
        //end get

    private:
        //data variable
        std::string movie_name;
        float score = 0; // default score for every movie is 0 unless user enter
any number for score
        unsigned int year = 1895; // default history for every movie is 1895 (tim
e of production of first movie) unless user enter any number for year
        unsigned int time = 0; // default time for every movie is 0 unless user e
nter any time for movie
        std::string director; // the name of director of movie
        std::string country; // name of production country of movie's
        unsigned int id = 0; // maximum random number expected 3000 and start at
0
        float ** score_details;
    };

#endif // !MOVIE_H
```

توابع set وظیفه validation و اعتبارسنجی برای قرار دادن مقادیر کاربر در متغیر های private کلاس را بر عهده دارند. این توابع هوشمند هستند و در برابر ورودی گرفته شده از کاربر ارور های متفاوتی را چاپ می کنند.

توابع get وظیفه گرفتن اطلاعات از کلاس برای چاپ کردن در تابع اصلی برنامه را بر عهده دارند.

12 تابع prototype شان در فایل func.hpp تعریف شده است که وظیفه های مختلف برنامه مانند گرفتن مقادیر از کاربر ، فراخوانی مناسب توابع کلاس ، چاپ ارور ، اعتبارسنجی ، چاپ خروجی ها و... را به عهده دارند.

FUNC.HPP

```
#ifndef FUNK_H
#define FUNK_H

void add(Movie, Movie * &, int &);
bool remove(std::string, Movie * &, int &);
void show(std::string, Movie * &, int &);
void show_all(Movie * &, int &);
void sort_by(Movie * &, std::string, int &, bool (*function)(Movie * &, std::stri
ng, int));
void avg_score(Movie *, int);
void help();
bool checkmovie(std::string, Movie *, int, int &); // check for available movie's
 in list
void score_details(std::string, Movie*, int &);
void order_adjust(std::string &);
void Command_Separator(std::string, Movie * &, int &);
bool help_sort(Movie * & array, std::string select, int j); // passing this funci
on to sort_by for sorting by name's or score's

#endif // !FUNK_H
```

که با تابع اصلی برنامه یعنی تابع main ، 29 تابع این برنامه را تشکیل می دهند که با سپردن وظایف به توابع و تعریف کلاس ، تابع main برنامه بسیار تمیز است.

در ادامه به عملکرد توابع برنامه می پردازیم.

Class Movie

```
void set_movie_name(std::string);
```

مقادیر وارد شده از طرف کاربر برای نام فیلم فقط باید شامل عدد و حروف باشد. مثل : Spiderman3

این تابع وظیفه وظیفه دارد تا چک کند که عبارت وارد شده توسط کاربر فقط شامل حروف و عدد باشد و در صورت اشکال در نام فیلم ارور مناسب را به کاربر نمایش می دهد. این تابع درون یک Try, Catch نوشته شده است و در هنگام بروز خطا، آن خطا را به خود تابع به اصطلاح پرتاب می کند.

Check_add یک متغیر Boolean و Global است که به طور پیشفرض true است و در صورت بروز هر گونه خطا در اعتبار سنجی های بخش های توابع set این متغیر به حالت false در می آید. سپس این متغیر به صورت extern به فایل func.cpp فرستاده می شود و در صورتی تابع add اجرا می شود که این متغیر true باشد و در هیچ مرحله ای از اعتبار سنجی دچار خطا نشده باشیم.

```
void Movie::set_movie_name(string tempname)
{
    for (size_t i = 0; tempname[i] != '\0' ; i++)
    {
        // name of movie be must included letters and numbers
        if (!isalnum(tempname[i]))
        {
            check_add = false;
        }
    }
    try{
        //validation for invalid argument and printing error
```

```
        if (!check_add)
        {
            throw invalid_argument("The name of movie be must included letters an
d numbers, you entered invalid character's!");
        }
        movie_name = tempname;
    }
    catch(invalid_argument & moviename){
        cerr << "ERROR invalid argument:" << endl << moviename.what() << endl;
    }
}
```

void set_score(float);

سیستم امتیاز دهی به فیلم ها در این برنامه مانند imdb طراحی شده است.

امتیاز یک عدد اعشاری بین 0 تا 10 است. این تابع این اعتبارسنجی را انجام می دهد و در صورت بروز خطا، پیغام مناسب را چاپ می کند.

```
void Movie::set_score(float tempscore)
{
    try{
        if (tempscore > 10 || tempscore < 0)
        {
            check_add = false;
            throw invalid_argument("Score number be must between 0 and 10, you en
terd invalid number!");
        }
        score = tempscore;
    }
    catch(invalid_argument & score)
    {
        cerr << "ERROR invalid argument:" << endl << score.what() << endl;
    }
}
```

void set_year(unsigned int);

اولین فیلم تاریخ سینما در سال 1895 ساخته شد و هم اکنون در سال 2021 هستیم. این تابع این اعتبارسنجی را انجام می دهد و اگر سال تولید فیلم در این رنج نباشد خطای مناسب را چاپ می کند.

```
void Movie::set_year(unsigned int tempyear)
{
    try{
        if (tempyear > 2021 || tempyear < 1895)
        {
            check_add = false;
            throw invalid_argument("History of production of first movie was 1895
 and now we are in 2021. So you must enter a year between 1895 and 2021 :)");
        }
        year = tempyear;
    }
    catch(invalid_argument & year){
        cerr << "ERROR invalid argument:" << endl << year.what() << endl;
    }
}
```

void set_time(unsigned int);

زمان یک فیلم نمی تواند 0 دقیقه یا عدد منفی ای باشد. این تابع وظیفه دارد که این اعتبارسنجی را انجام دهد و در صورت خطا پیغام مناسب به کاربر نمایش دهد.

```
void Movie::set_time(unsigned int temptime)
{
    try{
        if (temptime < 1)
        {
            check_add = false;
            throw invalid_argument("Time of movie be must longer than 0 min");
        }
        time = temptime;
    }
    catch(invalid_argument & time){
        cerr << "ERROR invalid argument:" << endl << time.what() << endl;
    }
```

```
}
```

void set_director(std::string);

نام کارگردان فیلم فقط می تواند شامل حروف باشد. وظیفه این تابع اعتبارسنجی برای این مورد است. در صورت عدم مشکل می تواند این مقداری که از کاربر گرفته است را در متغیر های خصوصی کلاس قرار می دهد.

```
void Movie::set_director(string directortemp)
{
    for (size_t i = 0; directortemp[i] != '\0' ; i++)
    {
        // name of movie be must included letters
        if (!isalpha(directortemp[i]))
        {
            check_add = false;
        }
    }
    try{
        // validation for invalid argument and printing error
        if (!check_add)
        {
            throw invalid_argument("The name of director of movie be must include
d letters, you entered invalid character's!");
        }
        director = directortemp;
    }
    catch(invalid_argument & director){
        cerr << "ERROR invalid argument:" << endl << director.what() << endl;
    }
}
```

void set_country(std::string);

اسم کشور سازنده یک فیلم فقط می تواند شامل حروف باشد.

```
void Movie::set_country(string countrytemp)
{
    for (size_t i = 0; countrytemp[i] != '\0' ; i++)
    {
        // name of country's movie be must included letters
        if (!isalpha(countrytemp[i]))
        {
            check_add = false;
        }
    }
    try{
        // validation for invalid argument and printing error
        if (!check_add)
        {
            throw invalid_argument("The name of country's production of movie be
must included letters, you entered invalid character's!");
        }
        country = countrytemp;
    }
    catch(invalid_argument & country){
        cerr << "ERROR invalid argument:" << endl << country.what() << endl;
    }
}
```

```
void set_random_id();
```

برای نسبت دادن یک شناسه (ID) به هر فیلم به کار می رود که باید یک عدد متمایز و رندوم باشد. روشی جایگزین default_random_engine نیز پیدا کردم که به صورت کامنت به برنامه اضافه شده است.

default_random_engine در بعضی از سیستم عامل ها مانند ویندوز به درستی کار نمی کند. البته در این تابع این جوانب سنجیده شده است و با استفاده از random_device این مشکل حل شده است. اما برای محکم کاری و پشتیبانی سریع تر، در صورت خطا در سیستم مورد نظر می توان کد های کامنت شده را استفاده کرد.

```cpp
void Movie::set_random_id()
{
    random_device dev;
    default_random_engine eng (dev());
    uniform_int_distribution <unsigned int> Myrand(1000, 3000);
    id = Myrand(eng);

    // this is better in random number ...

    // random_device dev;
    // std::mt19937 rng(dev());
    // uniform_int_distribution<int> Myrand(0, 1000000);

    // but i'am not using this because in this practice mentioned random engine

}
```

void set_score_details();

در این بخش ما یک آرایه دو بعدی پویا تعریف می کنیم که طول ابعاد آرایه توسط این تابع گرفته شده، آرایه ساخته می شود و مقادیر امتیاز های منتقدین در آن قرار می گیرد.

بعد اول آرایه تعداد منتقدین و بعد دوم تعداد امتیاز هایی است که هر منتقد به بخش ها مختلف فیلم می دهد.

```cpp
void Movie::set_score_details()
{
    unsigned int numberx = 0 , numbery = 0;
    cin >> numberx >> numbery;
    score_details = new float * [numberx];
    for (size_t i = 0; i < numberx; i++)
    {
        score_details[i] = new float [numbery];
    }
    for (size_t i = 0; i < numberx; i++)
    {
        for (size_t j = 0; j < numbery; j++)
        {
            cin >> score_details[i][j];
```

```
            if(score_details[i][j] < 0 && score_details[i][j > 100])
            {
                throw invalid_argument("Score's can be between 0 and 100, you ent
erd invalid number!");
            }
        }
    }
}
```

در توابع get این کلاس یا متغیر های ذخیره شده در آرایه به یک تابع دیگر برنامه برای چاپ شدن پاس می شوند و یا در همان تابع چاپ می شوند.

فایل هدر func.hpp

```
#ifndef FUNK_H
#define FUNK_H

void add(Movie, Movie * &, int &);
bool remove(std::string, Movie * &, int &);
void show(std::string, Movie * &, int &);
void show_all(Movie * &, int &);
void sort_by(Movie * &, std::string, int &, bool (*function)(Movie * &, std::stri
ng, int));
void avg_score(Movie *, int);
void help();
bool checkmovie(std::string, Movie *, int, int &); // check for available movie's
 in list
void score_details(std::string, Movie*, int &);
void order_adjust(std::string &);
void Command_Separator(std::string, Movie * &, int &);
bool help_sort(Movie * & array, std::string select, int j); // passing this funci
on to sort_by for sorting by name's or score's

#endif // !FUNK_H
```

```
void add(Movie, Movie * &, int &);
```

چون تابع add در برنامه زیاد استفاده می شود، به صورت inline نوشته شده است تا برنامه سرعت بیشتری داشته باشد.

در این تابع آرگون های آرایه و... به صورت call by refrence و با استفاده از & فرستاده شدند تا تغییراتی که روی آن ها اعمال می شود در سرتاسر برنامه وجود داشته باشد.

در این برنامه باید زمانی که فیلم sPiDerMAN اضافه می شود، اگر فیلم spiderman در آرایه وجود داشته باشد نباید اجازه اضافه کردن آن داده شود و برنامه نسبت به این مورد هوشمند است. این کار را تابع checkmovie انجام می دهد. در صورت عدم مشکل دیتاهایی که از کاربر گرفته و در temp ریخته شده است به آرایه پویا اضافه می کنیم و هر بار طول آرایه افزایش پیدا می کند.

```
inline void add(Movie temp, Movie * & array,  int & n)
{
    // for checking available moive's in list to adding name to list
    int i;
    string nameofmovie = temp.get_movie_name();
    if(checkmovie(nameofmovie, array, n, i))
    {
        throw invalid_argument("ERROR: The name with you enter is available in mo
vie's list, please enter another name for your movie");
    }

    // increase length of array for adding new element
    if(check_add)
    {
        Movie * newarray;
        newarray = new Movie[n + 1];
        for (int j = 0; j < n; j++)
        {
            newarray[j] = array[j];
        }
        newarray[n] = temp;
        array = new Movie[n + 1];
        for (int j = 0; j < n + 1; j++)
        {
```

```
            array[j] = newarray[j];
        }
        delete[] newarray;
        n++; // size of array in next call add's function be must assign size of
newarray
        cout << "--> "<< array[n-1].get_movie_name() << " added" << endl;
    }
}
```

```
bool remove(std::string, Movie * &, int &);
```

به این دلیل که برنامه نباید به حروف کوچک و بزرگ حساس باشد، پس در ابتدا از تابع checkmovie استفاده شده است. این تابع همه عناصر آرایه پویا را به یک آرایه موقت انتقال می دهد به جز آن فیلمی که کاربر قصد حذف کردن آن را دارد. سپس همه موارد موارد آرایه موقت را در آرایه اصلی کپی می کند و طول آرایه ها را یک واحد کم می کند.

در هر مرحله خطا ارور مناسب چاپ می شود.

```
void remove(string name_movie_selected, Movie *&array, int &n)
{
    if (n < 1)
    {
        throw invalid_argument("ERROR: no movies added!");
    }
    // for checking available moive's in list to adding name to list
    int i;
    if (!checkmovie(name_movie_selected, array, n, i))
    {
        throw invalid_argument("ERROR 404: your name's of movie that you entered,
 was not found!");
    }
    Movie *newarray;
    // because we are removing an element of array so we must be reduse size of a
rray (n - 1)
    newarray = new Movie[n - 1];
    int j, h = 0;
    for (j = 0; j < n; j++)
    {
```

```
        if (j != i)
        {
            newarray[h] = array[j];
            h++;
        }
    }
    array = new Movie[n - 1];
    for (j = 0; j < n - 1; j++)
    {
        array[j] = newarray[j];
    }
    delete[] newarray;
    n--; // length of array decrease
}
```

```
void show(std::string, Movie * &, int &);
```

اطلاعات فیلمی که کاربر می خواهد را چاپ می کند. در این تابع نیز برای پیدا کردن و جست و جو در آرایه به گونه که به حروف کوچک و بزرگ حساس نباشد از checkmovie استفاده شده است.

```
void show(string tempname, Movie *&array, int &n)
{
    int i = 0;
    if (checkmovie(tempname, array, n, i))
    {
        cout << "| " << setw(4) << array[i].get_id() << " | " << setw(12) << arra
y[i].get_movie_name() << " | " << setw(5) << array[i].get_score() << " | " << set
w(8) << array[i].get_director() << " | " << setw(4) << array[i].get_year() << " |
 " << setw(8) << array[i].get_country() << " | " << endl;
    }
    else
    {
        cerr << "ERROR: Your entered movie's name was not found!!!" << endl;
    }
}
```

```
void show_all(Movie * &, int &);
```

این تابع اطلاعات همه فیلم ها را نمایش می دهد. در واقع اسم فیلم ها به تابع show فرستاده می شود تا در هر مرحله چاپ شود.

```
void show_all(Movie *&array, int &n)
{
    bool check = false;
    cout << "| " << setw(4) << "ID"
         << " | " << setw(12) << "NAME"
         << " | " << setw(4) << "SCORE"
         << " | " << setw(8) << "DIRECTOR"
         << " | " << setw(4) << "YEAR"
         << " | " << setw(8) << "COUNTRY"
         << " | " << endl;

    for (size_t i = 0; i < n; i++)
    {
        check = true;
        show(array[i].get_movie_name(), array, n);
    }
    if (!check)
    {
        cerr << "ERROR: There are no movies to show!!!" << endl;
    }
}
```

```
void sort_by(Movie * &, std::string, int &, bool (*function)(Movie * &, std::string, int));
```

این تابع باید عناصر فیلم را بر اساس نام یا امتیاز آن ها مرتب کند. در این تابع از bubble sort بهینه استفاده شده است و یک تابع به عنوان آرگومان به آن پاس شده است که مشخص می کند که تابع باید بر چی اساسی مرتب شوند. نام یا امتیاز

```
void sort_by(Movie *&array, string select, int &n, bool (*function)(Movie *&, str
ing, int))
{
    bool check = true /* for optimizing bubble sort */, sort = false;
    if (n > 0)
    {
        try
        {
            for (size_t i = 0; i < n && check; i++)
            {
                check = false;
                for (size_t j = 0; j < n - i - 1; j++)
                {
                    if (function(array, select, j))
                    {
                        Movie temp = array[j];
                        array[j] = array[j + 1];
                        array[j + 1] = temp;
                        check = true;
                        sort = true;
                    }
                }
            }
        }
        catch (invalid_argument &errorselected)
        {
            cerr << "ERROR: " << errorselected.what() << " you entered \"" << sel
ect << "\"!" << endl;
        }
    }
    else
    {
        cerr << "--> ERROR: " << endl
             << "No exist any movies for sorting!" << endl;
    }
    if (sort)
    {
        cout << "--> Movies sorted by " << select << endl;
    }
}
```

```
bool help_sort(Movie * & array, std::string select, int j);
```

این تابع به عنوان آرگومان ورودی به تابع sort_by فرستاده می شود. وظیفه آن این است که تشخیص دهد مرتب سازی باید بر چه اساسی صورت بگیرد و اگر مشکلی در داده های ارسالی بود، ارور مناسب را نمایش دهد.

```
bool help_sort(Movie *&array, string select, int j)
{
    if (select == "name")
    {
        if (array[j].get_movie_name() > array[j + 1].get_movie_name())
        {
            return true;
        }
        return false;
    }
    else if (select == "score")
    {
        if (array[j].get_score() < array[j + 1].get_score())
        {
            return true;
        }
        return false;
    }
    else
    {
        throw invalid_argument("ERROR: Movies only can order by name's or score's
!");
    }
    return false;
}
```

void avg_score(Movie *, int);

این تابع میانگین امتیاز اصلی همه فیلم هایی که در آرایه وجود دارد را بدست می آورد.

```
void avg_score(Movie *array, int n)
{
    if (n > 0)
    {
        float avg = 0;
        for (size_t i = 0; i <= n; i++)
        {
            avg += array[i].get_score();
        }
        float result = avg / n;
        cout << "Average Score: " << result << endl;
    }
    else
    {
        throw invalid_argument("No moives added!");
    }
}
```

void help();

این تابع نحوه نوشتن دستورات در این برنامه را به کاربر نمایش می دهد و یک راهنمای کلی برای کاربران برای استفاده صحیح از این برنامه است.

```
void help()
{
    cout << endl
         << "-----------------------------------" << endl;
    cout << "Add Movie: "
         << "add < movie name > < score > < production Year > < directory name >
< time > < production country > " << endl;
    cout << "--> Gauide:" << endl
```

```
        << "The name of movie be must included letters and numbers" << endl
        << "Score number be must between 0 and 10" << endl
        << "History of production of first movie was 1895 and now we are in 2021
. So you must enter a year between 1895 and 2021" << endl
        << "Time of movie be must longer than 0 min" << endl
        << "The name of director of movie be must included letters, you entered
invalid character's!" << endl
        << "The name of country's production of movie be must included letters"
        << "--> Example : add SpiderMan3 10 2014 nolan 120 Iran"
        << endl
        << endl;

    cout << "Remove Movie: "
        << "remove < movie name >" << endl
        << "--> Example: remove SpiderMan3"
        << endl
        << endl;

    cout << "Sort Movie's: "
        << "sort_by < score/name >" << endl;
    cout << "--> Gauide:" << endl
        << "Movies can sorted by \" name \" or \" score \""
        << "--> Example: sort_by name"
        << endl
        << endl;

    cout << "Show Movie: "
        << "show < movie name >" << endl
        << "--> Gauide:" << endl
        << "showing data of a movie in a line"
        << "--> Example: show SpiderMan3"
        << endl
        << endl;

    cout << "Show All Movie's: "
        << "show-all" << endl
        << "--> Gauide:" << endl
        << "showing data's of all movie with added !" << endl
        << "--> Example:  show-all"
        << "--> be attention to - in order of show-all"
        << endl
        << endl;

    cout << "Average Score Of Movie's: "
        << "average-score" << endl
        << "--> Gauide:" << endl
        << "calculates average score of all of movies with added" << endl
        << "--> Example: average-score "
```

```
         << endl
         << endl;

    cout << "Score Details Of Movie's: "
         << "score-details" << endl
         << "--> Gauide:" << endl
         << "score details get two argument. get and set" << endl
         << "set getting two number. first number is number of reviewer and secon
d number is score's of them!" << endl
         << "and next enter score's" << endl
         << "get getting two number. first number is number of reviewer and secon
d number is score's number" << endl
         << "--> Example:" << endl
         << "score-details spIderMan3 set 3 2 100 85 80 73 60 70" << endl
         << "score-details SpiderMan3 get 2 2 \t"
         << " print --> 73" << endl;

    cout << "Help: "
         << "help" << endl
         << "--> Gauide:" << endl
         << "showing this list"
         << endl
         << endl;
    cout << "-----------------------------------" << endl;
}
```

```
bool checkmovie(std::string, Movie *, int, int &);
```

این تابع برای حساس نبودن برنامه به حروف بزرگ و کوچک نوشته شده است. قطعا طول دو عبارت spiderman و SpIdeRMAN برابر است و مشکل در بزرگ و کوچک بودن آن هاست. پس شرط برابری طول برای بهینه سازی تابع گذاشته شده است.

در صورتی که طول برابر باشد، اسم فیلم مورد نظر و نامی که کاربر وارد کرده را تماما به حروف کوچک تبدیل می کند و سپس با هم مقایسه می کند.

```
bool checkmovie(string temp, Movie *array, int n, int &i /* in show function we u
se i for showing data */)
{
    for (i = 0; i < n; i++)
    {
        if (temp.length() == array[i].get_movie_name().length()) // if name's wer
e similar length of them is equal
        {
            string tempname = temp;
            string namemovie = array[i].get_movie_name();
            for (size_t q = 0; tempname[q] != '\0'; q++)
            {
                if (isupper(tempname[q]))
                {
                    tempname[q] = tolower(tempname[q]);
                }
                if (isupper(namemovie[q]))
                {
                    namemovie[q] = tolower(namemovie[q]);
                }
            }
            if (tempname == namemovie)
            {
                return true;
            }
        }
    }
    return false;
}
```

```
void order_adjust(std::string &);
```

این تابع برای حساس نبودن دستورات برنامه مانند add, remove و... به حروف کوچک و بزرگ نوشته شده است. دستورات را تماما به حروف کوچک تبدیل می کند و سپس تابع Command_Separator از دستورات استفاده می کند.

```
void order_adjust(string &select)
{
    // for no sensitive select to lower or upper case
    for (size_t q = 0; select[q] != '\0'; q++)
```

```
    {
        if (isupper(select[q]))
        {
            select[q] = tolower(select[q]);
        }
    }
}
```

```
void Command_Separator(std::string, Movie * &, int &);
```

این تابع وظیفه دارد که دستوراتی که کاربر وارد می کند را جدا کند و با توجه به دستورات وارد شده، کاربر را هدایت کند.

```
void Command_Separator(string select, Movie *&array, int &n)
{
    order_adjust(select);

    if (select == "add")
    {
        Movie film;
        string tempname, directortemp, countrytemp;
        unsigned int temptime, tempyear;
        float tempscore;
        cin >> tempname >> tempscore >> tempyear >> directortemp >> temptime >> c
ountrytemp;
        film.set_movie_name(tempname);
        film.set_score(tempscore);
        film.set_director(directortemp);
        film.set_year(tempyear);
        film.set_country(countrytemp);
        film.set_random_id();
        // Movie film(tempname, tempscore, tempyear, directortemp, temptime, coun
trytemp);

        // cout << tempname << "\t" << tempscore << "\t" << directortemp << "\t"
<< tempyear << "\t" << countrytemp << endl;
        try
        {
            add(film, array, n);
        }
```

```
        catch (invalid_argument &nameofmovie)
        {
            cerr << "ERROR: " << nameofmovie.what() << endl;
        }
    }
    else if (select == "remove")
    {
        string tempname;
        cin >> tempname;
        try
        {
            remove(tempname, array, n);
            cout << tempname << " Removed!" << endl;
        }
        catch (invalid_argument &remove)
        {
            cerr << remove.what() << endl;
        }
    }
    else if (select == "sort-by")
    {
        string select;
        cin >> select;
        sort_by(array, select, n, &help_sort);
    }
    else if (select == "show")
    {
        string tempname;
        cin >> tempname;
        show(tempname, array, n);
    }
    else if (select == "show-all")
    {
        show_all(array, n);
    }
    else if (select == "average-score")
    {
        try
        {
            avg_score(array, n);
        }
        catch (invalid_argument &avg)
        {
            cerr << "ERROR: " << avg.what() << endl;
        }
    }
    else if (select == "help")
    {
```

```
        help();
    }
    else if (select == "score-details")
    {
        string tempname;
        cin >> tempname;
        score_details(tempname, array, n);
    }
    else
    {
        cerr << "ERROR input data" << endl
             << "You Enter: " << select << endl
             << "Plsase Enter \" help \" to show manuals of program! " << endl;
    }
}
```

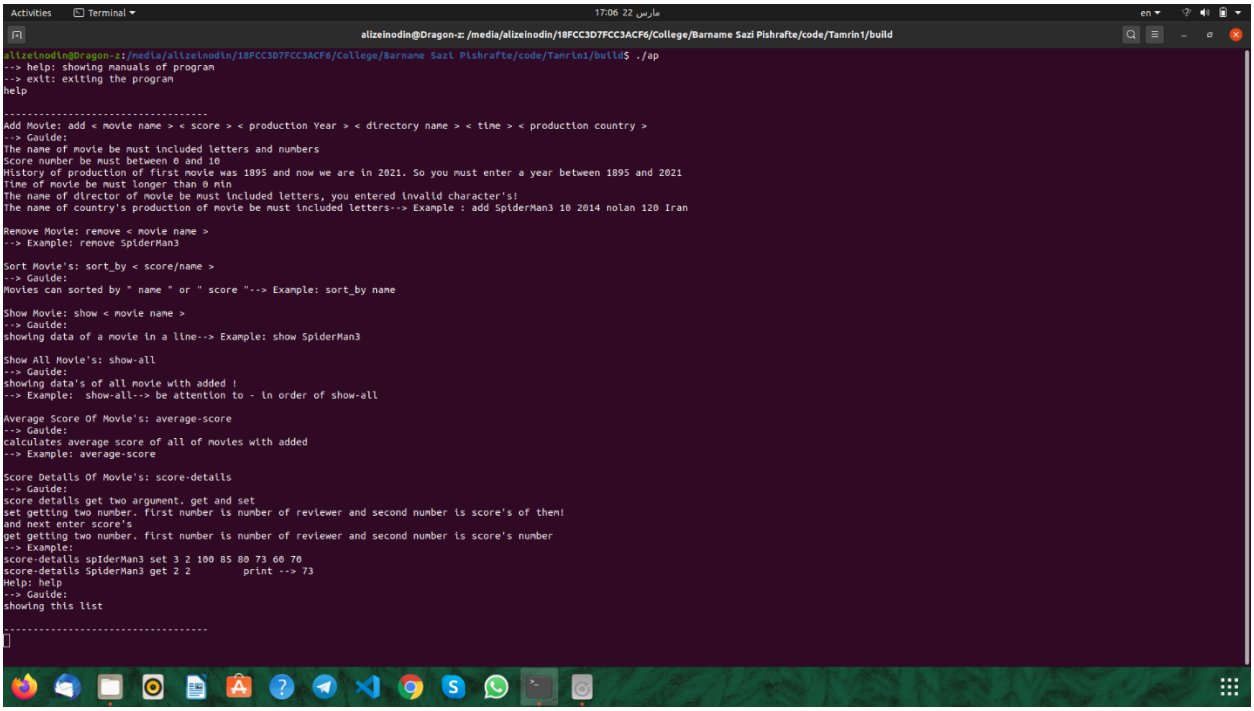
```
void score_details(std::string, Movie*, int &);
```

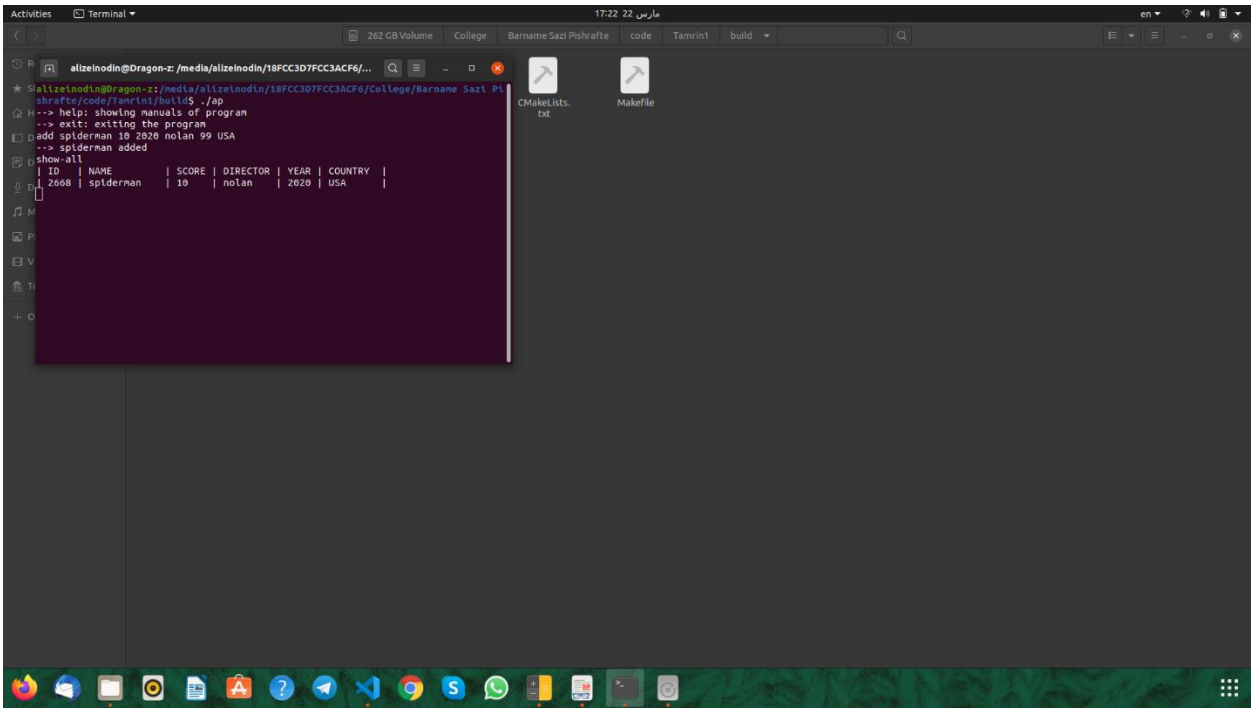
این تابع داده های مرتبط به امتیاز های مختلف منتقدان را از کاربر دریافت می کند و آن را به توابع set و get بخش score_details هدایت می کند.

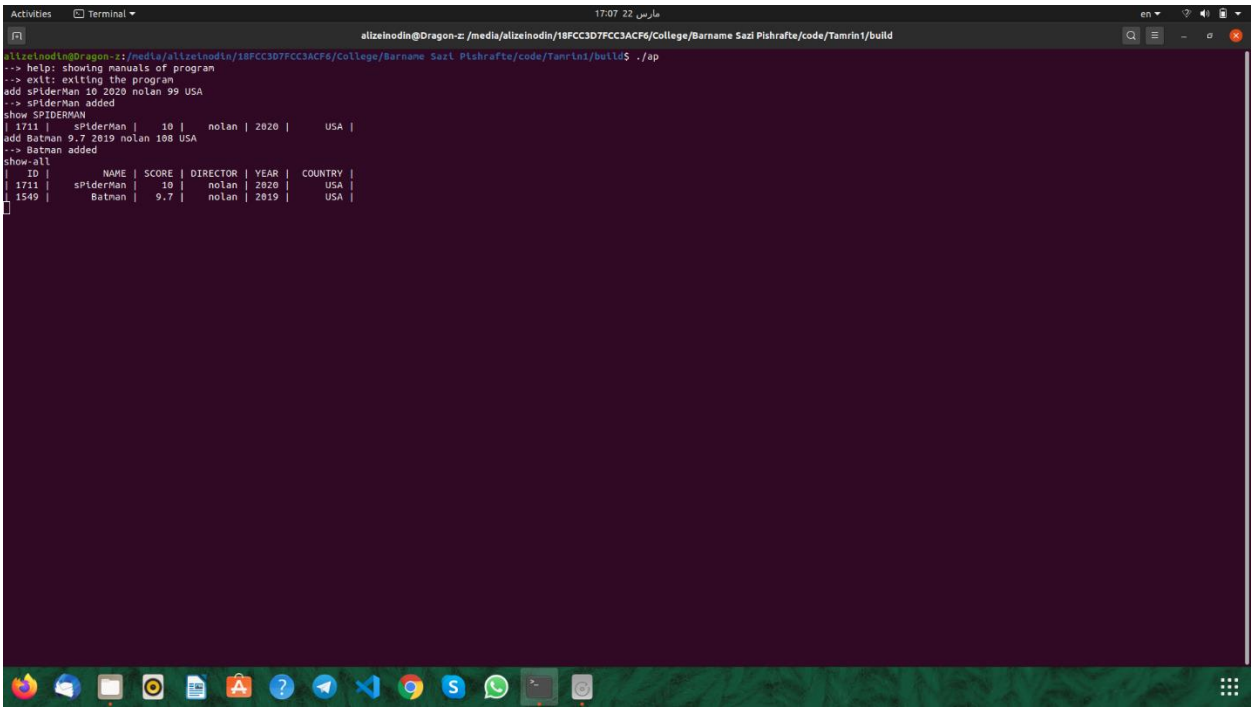
```
void score_details(string tempname, Movie *array, int &n)
{
    int i = 0;
    if (!checkmovie(tempname, array, n, i))
    {
        throw invalid_argument("ERROR: The name with you entered is not available
 in list of movie's!, please enter a valid name");
    }
    string select_score;
    cin >> select_score;
    order_adjust(select_score);
    if (select_score == "set")
    {
        array[i].set_score_details();
        cout << "Score's added for " << array[i].get_movie_name() << endl;
    }
    else if (select_score == "get")
    {
        cout << "Score 2 & 2 for " << array[i].get_movie_name() << " --> ";
        array[i].get_score_details();
    }
    else
```

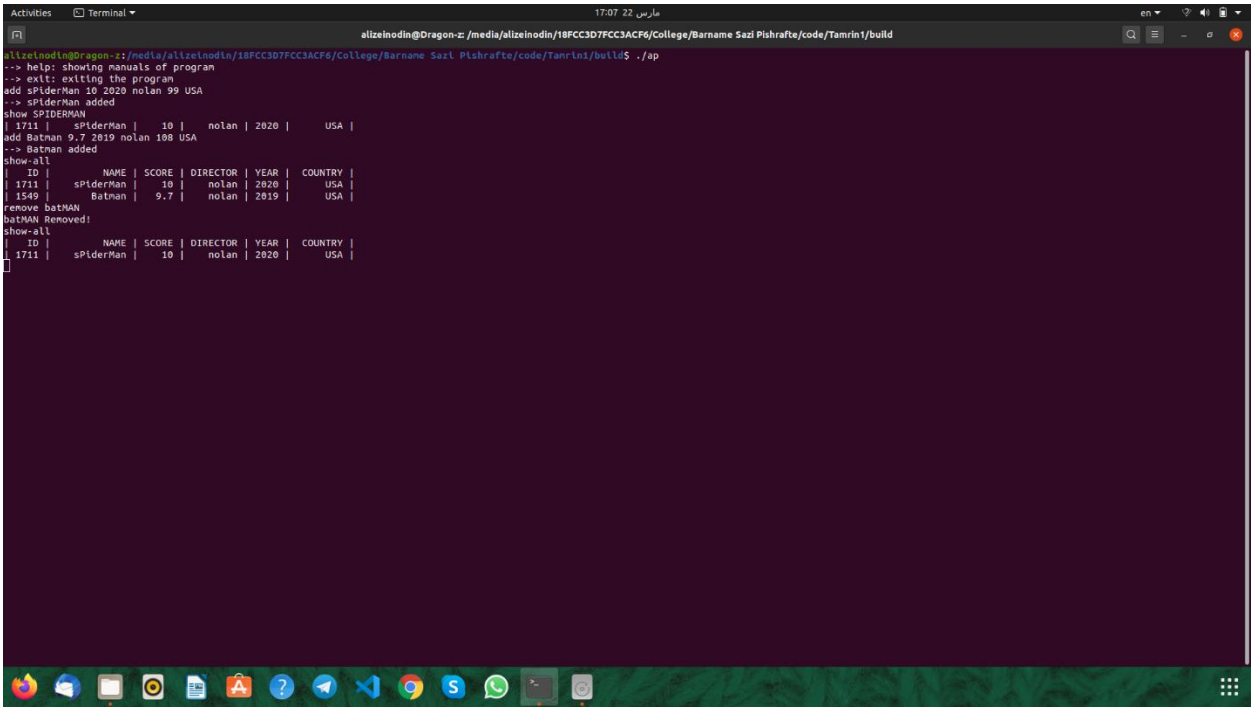
```
    {
        throw invalid_argument("ERROR: You enter invalid argument. Score-
Details can get two value, get and set!");
    }
}
```

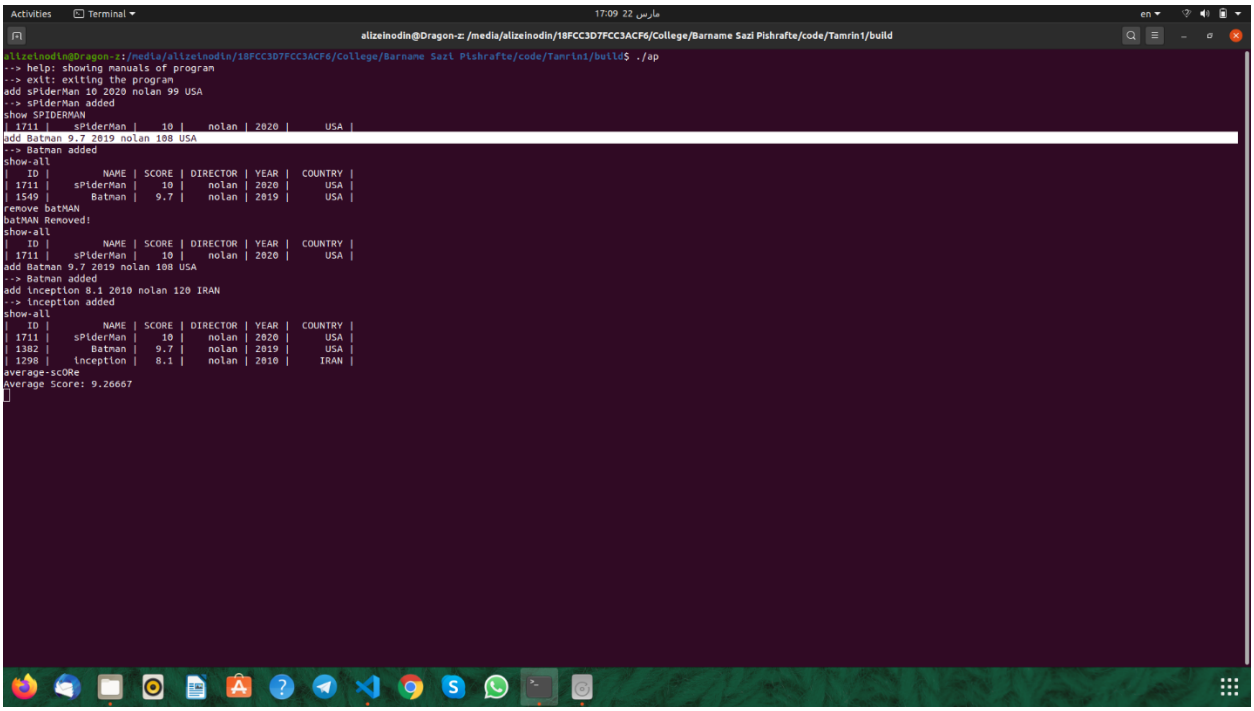
خروجی برنامه

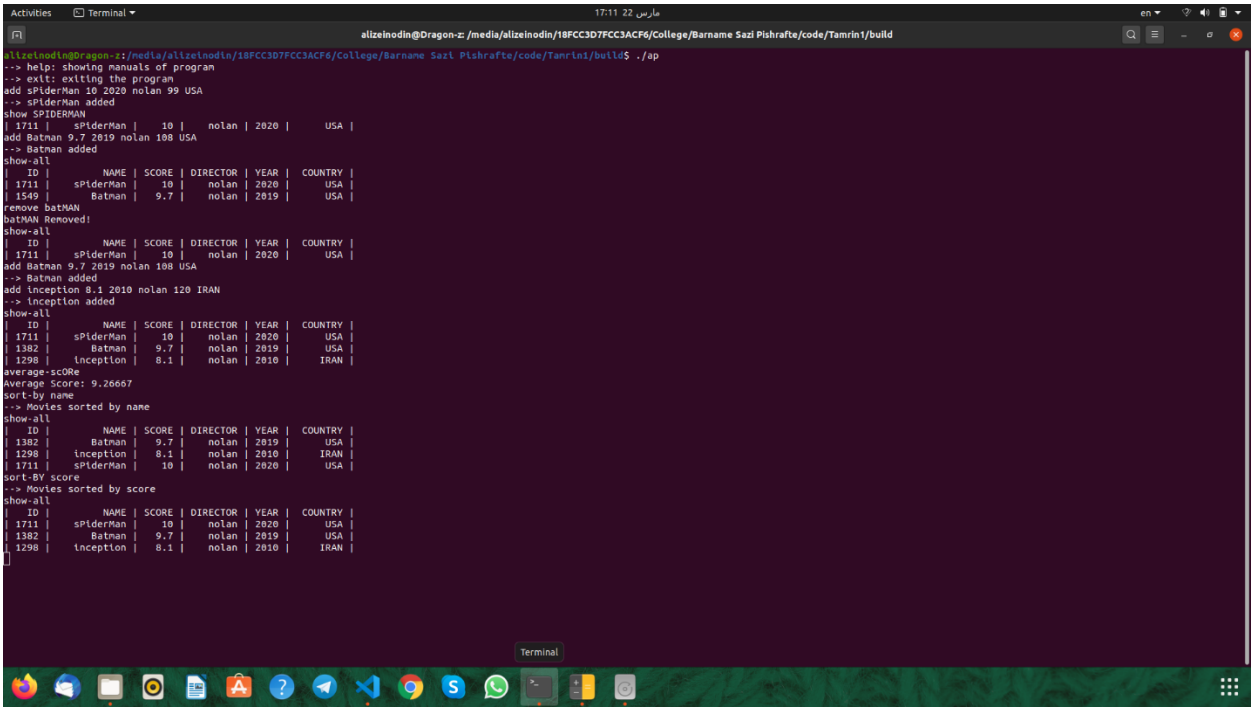


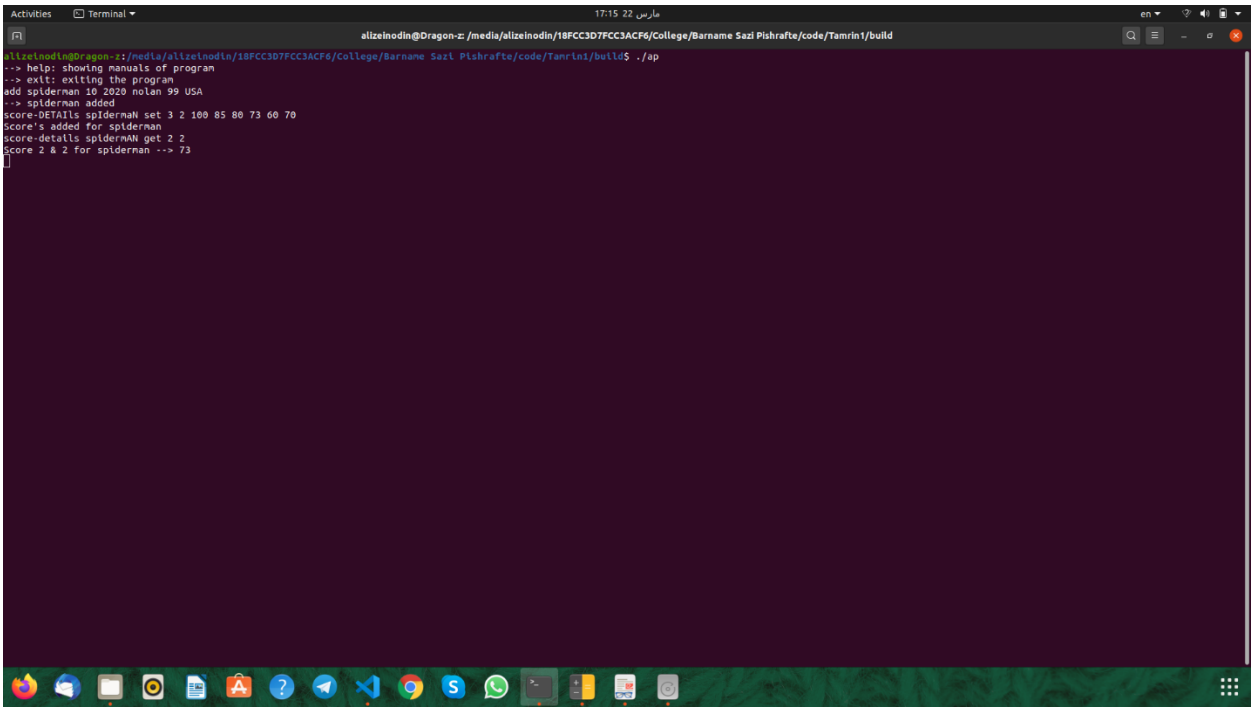


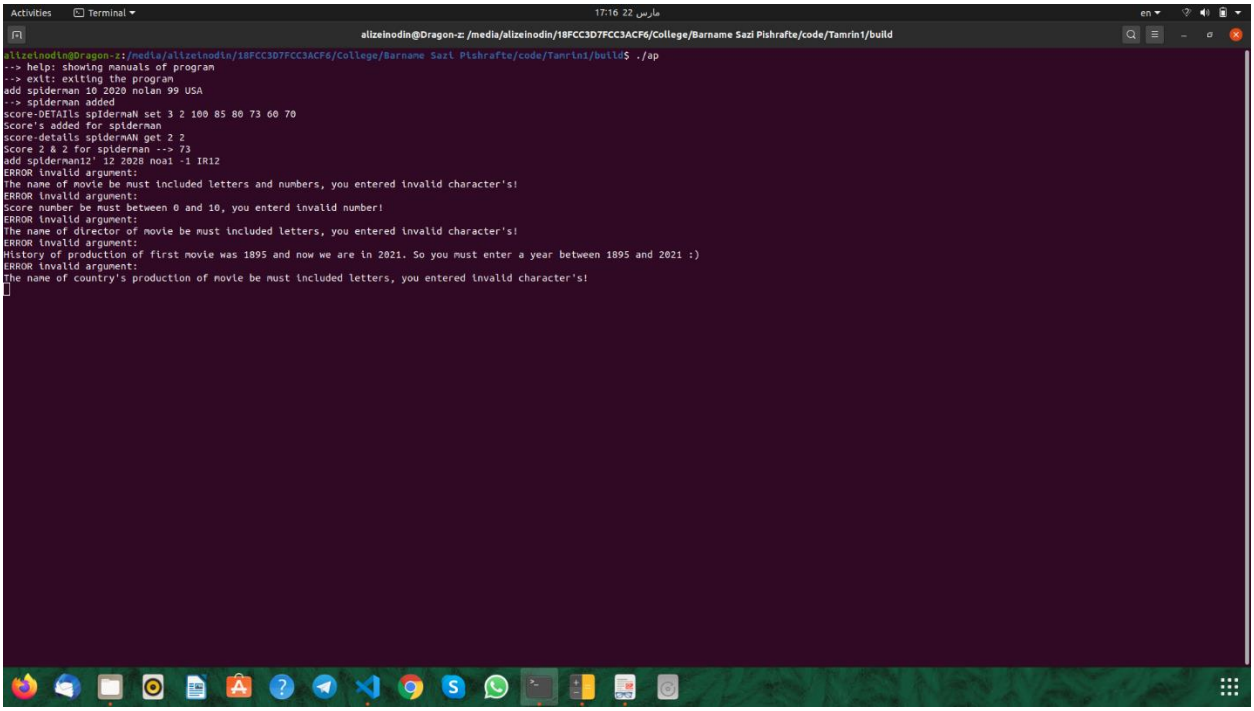

Activities
Terminal
17:07 22 مارس
en
alizeinodin@Dragon-z: /media/alizeinodin/18FCC3D7FCC3ACF6/College/Barname Sazi Pishrafte/code/Tamrin1/build
alizeinodin@Dragon-z:/media/alizeinodin/18FCC3D7FCC3ACF6/College/Barname Sazi Pishrafte/code/Tamrin1/build$ ./ap
--> help: showing manuals of program
--> exit: exiting the program
add sPiderMan 10 2020 nolan 99 USA
--> sPiderMan added
show SPIDERMAN
| 1711 | sPiderMan | 10 | nolan | 2020 | USA |
add Batman 9.7 2019 nolan 108 USA
--> Batman added
show-all
ID	NAME	SCORE	DIRECTOR	YEAR	COUNTRY
1711	sPiderMan	10	nolan	2020	USA
1549	Batman	9.7	nolan	2019	USA


Activities
Terminal
17:07 22 مارس
en
alizeinodin@Dragon-z: /media/alizeinodin/18FCC3D7FCC3ACF6/College/Barname Sazi Pishrafte/code/Tamrin1/build
alizeinodin@Dragon-z:/media/alizeinodin/18FCC3D7FCC3ACF6/College/Barname Sazi Pishrafte/code/Tamrin1/build$ ./ap
--> help: showing manuals of program
--> exit: exiting the program
add sPiderMan 10 2020 nolan 99 USA
--> sPiderMan added
show SPIDERMAN
| 1711 | sPiderMan | 10 | nolan | 2020 | USA |
add Batman 9.7 2019 nolan 108 USA
--> Batman added
show-all
ID	NAME	SCORE	DIRECTOR	YEAR	COUNTRY
1711	sPiderMan	10	nolan	2020	USA
1549	Batman	9.7	nolan	2019	USA
remove batMAN
batMAN Removed!
show-all
ID	NAME	SCORE	DIRECTOR	YEAR	COUNTRY
1711	sPiderMan	10	nolan	2020	USA


Activities
Terminal
17:09 22 مارس
en
alizeinodin@Dragon-z: /media/alizeinodin/18FCC3D7FCC3ACF6/College/Barname Sazi Pishrafte/code/Tamrin1/build
alizeinodin@Dragon-z:/media/alizeinodin/18FCC3D7FCC3ACF6/College/Barname Sazi Pishrafte/code/Tamrin1/build$ ./ap
--> help: showing manuals of program
--> exit: exiting the program
add sPiderMan 10 2020 nolan 99 USA
--> sPiderMan added
show SPIDERMAN
| 1711 | sPiderMan | 10 | nolan | 2020 | USA |
add Batman 9.7 2019 nolan 108 USA
--> Batman added
show-all
ID	NAME	SCORE	DIRECTOR	YEAR	COUNTRY
1711	sPiderMan	10	nolan	2020	USA
1549	Batman	9.7	nolan	2019	USA
remove batMAN
batMAN Removed!
show-all
ID	NAME	SCORE	DIRECTOR	YEAR	COUNTRY
1711	sPiderMan	10	nolan	2020	USA
add Batman 9.7 2019 nolan 108 USA
--> Batman added
add inception 8.1 2010 nolan 120 IRAN
--> inception added
show-all
ID	NAME	SCORE	DIRECTOR	YEAR	COUNTRY
1711	sPiderMan	10	nolan	2020	USA
1382	Batman	9.7	nolan	2019	USA
1298	inception	8.1	nolan	2010	IRAN
average-scORe
Average Score: 9.26667


Activities
Terminal
17:11 22 مارس
en
alizeinodin@Dragon-z: /media/alizeinodin/18FCC3D7FCC3ACF6/College/Barname Sazi Pishrafte/code/Tamrin1/build
alizeinodin@Dragon-z:/media/alizeinodin/18FCC3D7FCC3ACF6/College/Barname Sazi Pishrafte/code/Tamrin1/build$ ./ap
--> help: showing manuals of program
--> exit: exiting the program
add sPiderMan 10 2020 nolan 99 USA
--> sPiderMan added
show SPIDERMAN
| 1711 | sPiderMan | 10 | nolan | 2020 | USA |
add Batman 9.7 2019 nolan 108 USA
--> Batman added
show-all
ID	NAME	SCORE	DIRECTOR	YEAR	COUNTRY
1711	sPiderMan	10	nolan	2020	USA
1549	Batman	9.7	nolan	2019	USA
remove batMAN
batMAN Removed!
show-all
ID	NAME	SCORE	DIRECTOR	YEAR	COUNTRY
1711	sPiderMan	10	nolan	2020	USA
add Batman 9.7 2019 nolan 108 USA
--> Batman added
add inception 8.1 2010 nolan 120 IRAN
--> inception added
show-all
ID	NAME	SCORE	DIRECTOR	YEAR	COUNTRY
1711	sPiderMan	10	nolan	2020	USA
1382	Batman	9.7	nolan	2019	USA
1298	inception	8.1	nolan	2010	IRAN
average-scORe
Average Score: 9.26667
sort-by name
--> Movies sorted by name
show-all
ID	NAME	SCORE	DIRECTOR	YEAR	COUNTRY
1382	Batman	9.7	nolan	2019	USA
1298	inception	8.1	nolan	2010	IRAN
1711	sPiderMan	10	nolan	2020	USA
sort-BY score
--> Movies sorted by score
show-all
ID	NAME	SCORE	DIRECTOR	YEAR	COUNTRY
1711	sPiderMan	10	nolan	2020	USA
1382	Batman	9.7	nolan	2019	USA
1298	inception	8.1	nolan	2010	IRAN
Terminal


Activities
Terminal
17:15 22 مارس
en
alizeinodin@Dragon-z: /media/alizeinodin/18FCC3D7FCC3ACF6/College/Barname Sazi Pishrafte/code/Tamrin1/build
alizeinodin@Dragon-z:/media/alizeinodin/18FCC3D7FCC3ACF6/College/Barname Sazi Pishrafte/code/Tamrin1/build$ ./ap
--> help: showing manuals of program
--> exit: exiting the program
add spiderman 10 2020 nolan 99 USA
--> spiderman added
score-DETAILs spIdermaN set 3 2 100 85 80 73 60 70
Score's added for spiderman
score-details spidermAN get 2 2
Score 2 & 2 for spiderman --> 73


Activities
Terminal
17:16 22 مارس
en
alizeinodin@Dragon-z: /media/alizeinodin/18FCC3D7FCC3ACF6/College/Barname Sazi Pishrafte/code/Tamrin1/build
alizeinodin@Dragon-z:/media/alizeinodin/18FCC3D7FCC3ACF6/College/Barname Sazi Pishrafte/code/Tamrin1/build$ ./ap
--> help: showing manuals of program
--> exit: exiting the program
add spiderman 10 2020 nolan 99 USA
--> spiderman added
score-DETAILs spIdermaN set 3 2 100 85 80 73 60 70
Score's added for spiderman
score-details spidermAN get 2 2
Score 2 & 2 for spiderman --> 73
add spiderman12' 12 2028 noa1 -1 IR12
ERROR invalid argument:
The name of movie be must included letters and numbers, you entered invalid character's!
ERROR invalid argument:
Score number be must between 0 and 10, you enterd invalid number!
ERROR invalid argument:
The name of director of movie be must included letters, you entered invalid character's!
ERROR invalid argument:
History of production of first movie was 1895 and now we are in 2021. So you must enter a year between 1895 and 2021 :)
ERROR invalid argument:
The name of country's production of movie be must included letters, you entered invalid character's!

MAIN.CPP

```cpp
#include"movie.hpp"
#include"func.hpp"
using namespace std;

int main() {

    int n = 0; // if user adding a new movie, in add function length of array wil
l increse but for saving space i try consider least space
    Movie * array;
    array = new Movie[n];
    string select;
    cout << "--> help: showing manuals of program" << endl
    << "--> exit: exiting the program" << endl;
    while (select != "exit")
    {
        cin >> select;
        if (select == "exit")
        {
            break; // doing not enter the program to command_separator with "exit
"
        }
        Command_Separator(select, array, n);
    }


    delete[] array;
    return 0;
}
```

در تابع اصلی برنامه یک آرایه پویا برای ذخیره سازی اطلاعاتی که کاربر می دهد در نظر گرفته شده است که هربار که از add و remove استفاده شود، این طول تغییر می کند.

MOVIE.CPP

```cpp
#include "movie.hpp"
using namespace std;

bool check_add = true; // for checking all add option is successful

// Movie::Movie(string tempname, unsigned int tempscore, unsigned int tempyear, u
nsigned int temptime, string directortemp, string countrytemp)
// {
//     set_movie_name(tempname);
//     set_score(tempscore);
//     set_director(directortemp);
//     set_year(tempyear);
//     set_country(countrytemp);
//     set_random_id();
// }

// getting and validation name of movie start
void Movie::set_movie_name(string tempname)
{
    for (size_t i = 0; tempname[i] != '\0' ; i++)
    {
        // name of movie be must included letters and numbers
        if (!isalnum(tempname[i]))
        {
            check_add = false;
        }
    }
    try{
        //validation for invalid argument and printing error
        if (!check_add)
        {
            throw invalid_argument("The name of movie be must included letters an
d numbers, you entered invalid character's!");
        }
        movie_name = tempname;
    }
    catch(invalid_argument & moviename){
        cerr << "ERROR invalid argument:" << endl << moviename.what() << endl;
    }
}
// end

// for getting movie's name for print in main()
string Movie::get_movie_name() const
{
```

```
    return movie_name;
}
// getting and validation score of movie start
void Movie::set_score(float tempscore)
{
    try{
        if (tempscore > 10 || tempscore < 0)
        {
            check_add = false;
            throw invalid_argument("Score number be must between 0 and 10, you en
terd invalid number!");
        }
        score = tempscore;
    }
    catch(invalid_argument & score)
    {
        cerr << "ERROR invalid argument:" << endl << score.what() << endl;
    }
}
// end

// for getitng movie's score for print in main()
float Movie::get_score() const
{
    return score;
}
// getting and validation for history of movie start
void Movie::set_year(unsigned int tempyear)
{
    try{
        if (tempyear > 2021 || tempyear < 1895)
        {
            check_add = false;
            throw invalid_argument("History of production of first movie was 1895
 and now we are in 2021. So you must enter a year between 1895 and 2021 :)");
        }
        year = tempyear;
    }
    catch(invalid_argument & year){
        cerr << "ERROR invalid argument:" << endl << year.what() << endl;
    }
}
// end

// for getting movie's year for print in main()
unsigned int Movie::get_year() const
{
    return year;
```

```
}

// getting and validation for time of movie start
void Movie::set_time(unsigned int temptime)
{
    try{
        if (temptime < 1)
        {
            check_add = false;
            throw invalid_argument("Time of movie be must longer than 0 min");
        }
        time = temptime;
    }
    catch(invalid_argument & time){
        cerr << "ERROR invalid argument:" << endl << time.what() << endl;
    }
}
// end

// for getting movie's time for print in main()
unsigned int Movie::get_time() const
{
    return time;
}

// getting and validation for name of director of movie start
void Movie::set_director(string directortemp)
{
    for (size_t i = 0; directortemp[i] != '\0' ; i++)
    {
        // name of movie be must included letters
        if (!isalpha(directortemp[i]))
        {
            check_add = false;
        }
    }
    try{
        // validation for invalid argument and printing error
        if (!check_add)
        {
            throw invalid_argument("The name of director of movie be must include
d letters, you entered invalid character's!");
        }
        director = directortemp;
    }
    catch(invalid_argument & director){
        cerr << "ERROR invalid argument:" << endl << director.what() << endl;
    }
```

```
}
// end

// for getting name of director for print in main()
string Movie::get_director() const
{
    return director;
}

// getting and validation for name of country's production of movie start
void Movie::set_country(string countrytemp)
{
    for (size_t i = 0; countrytemp[i] != '\0' ; i++)
    {
        // name of country's movie be must included letters
        if (!isalpha(countrytemp[i]))
        {
            check_add = false;
        }
    }
    try{
        // validation for invalid argument and printing error
        if (!check_add)
        {
            throw invalid_argument("The name of country's production of movie be
must included letters, you entered invalid character's!");
        }
        country = countrytemp;
    }
    catch(invalid_argument & country){
        cerr << "ERROR invalid argument:" << endl << country.what() << endl;
    }
}
// end

// for getting name of country of movie for print in main()
string Movie::get_country() const
{
    return country;
}
// random id start
void Movie::set_random_id()
{
    random_device dev;
    default_random_engine eng (dev());
    uniform_int_distribution <unsigned int> Myrand(1000, 3000);
    id = Myrand(eng);
```

```
    // this is better in random number ...

    // random_device dev;
    // std::mt19937 rng(dev());
    // uniform_int_distribution<int> Myrand(0, 1000000);

    // but i'am not using this because in this practice mentioned random engine

}
// end

// for getting id of movie for print in main()
unsigned int Movie::get_id() const
{
    return id;
}
void Movie::set_score_details()
{
    unsigned int numberx = 0 , numbery = 0;
    cin >> numberx >> numbery;
    score_details = new float * [numberx];
    for (size_t i = 0; i < numberx; i++)
    {
        score_details[i] = new float [numbery];
    }
    for (size_t i = 0; i < numberx; i++)
    {
        for (size_t j = 0; j < numbery; j++)
        {
            cin >> score_details[i][j];
            if(score_details[i][j] < 0 && score_details[i][j > 100])
            {
                throw invalid_argument("Score's can be between 0 and 100, you ent
erd invalid number!");
            }
        }
    }
}
void Movie::get_score_details()
{
    unsigned int i, j;
    cin >> i >> j;
    cout << score_details[i-1][j-1] << endl;
}
```

FUNC.CPP

```cpp
#include "movie.hpp"
#include "func.hpp"
using namespace std;
extern bool check_add;

// add new movie
// ----------------------------
inline void add(Movie temp, Movie *&array, int &n)
{
    // for checking available moive's in list to adding name to list
    int i;
    string nameofmovie = temp.get_movie_name();
    if (checkmovie(nameofmovie, array, n, i))
    {
        throw invalid_argument("ERROR: The name with you enter is available in mo
vie's list, please enter another name for your movie");
    }

    // increase length of array for adding new element
    if (check_add)
    {
        Movie *newarray;
        newarray = new Movie[n + 1];
        for (int j = 0; j < n; j++)
        {
            newarray[j] = array[j];
        }
        newarray[n] = temp;
        array = new Movie[n + 1];
        for (int j = 0; j < n + 1; j++)
        {
            array[j] = newarray[j];
        }
        delete[] newarray;
        n++; // size of array in next call add's function be must assign size of
newarray
        cout << "--> " << array[n - 1].get_movie_name() << " added" << endl;
    }
}
// ----------------------------

// remove movies
// ----------------------------
void remove(string name_movie_selected, Movie *&array, int &n)
{
```

```
    if (n < 1)
    {
        throw invalid_argument("ERROR: no movies added!");
    }
    // for checking available moive's in list to adding name to list
    int i;
    if (!checkmovie(name_movie_selected, array, n, i))
    {
        throw invalid_argument("ERROR 404: your name's of movie that you entered,
 was not found!");
    }
    Movie *newarray;
    // because we are removing an element of array so we must be reduse size of a
rray (n - 1)
    newarray = new Movie[n - 1];
    int j, h = 0;
    for (j = 0; j < n; j++)
    {
        if (j != i)
        {
            newarray[h] = array[j];
            h++;
        }
    }
    array = new Movie[n - 1];
    for (j = 0; j < n - 1; j++)
    {
        array[j] = newarray[j];
    }
    delete[] newarray;
    n--; // length of array decrease
}
// ----------------------------

// for showing a movies details
// ----------------------------
void show(string tempname, Movie *&array, int &n)
{
    int i = 0;
    if (checkmovie(tempname, array, n, i))
    {
        cout << "| " << setw(4) << array[i].get_id() << " | " << setw(12) << arra
y[i].get_movie_name() << " | " << setw(5) << array[i].get_score() << " | " << set
w(8) << array[i].get_director() << " | " << setw(4) << array[i].get_year() << " |
 " << setw(8) << array[i].get_country() << " | " << endl;
    }
    else
    {
```

```
        cerr << "ERROR: Your entered movie's name was not found!!!" << endl;
    }
}
// ----------------------------

// for showing all movies details
// ----------------------------
void show_all(Movie *&array, int &n)
{
    bool check = false;
    cout << "| " << setw(4) << "ID"
         << " | " << setw(12) << "NAME"
         << " | " << setw(4) << "SCORE"
         << " | " << setw(8) << "DIRECTOR"
         << " | " << setw(4) << "YEAR"
         << " | " << setw(8) << "COUNTRY"
         << " | " << endl;

    for (size_t i = 0; i < n; i++)
    {
        check = true;
        show(array[i].get_movie_name(), array, n);
    }
    if (!check)
    {
        cerr << "ERROR: There are no movies to show!!!" << endl;
    }
}
// ----------------------------

// for sorting array
// ----------------------------

// helping to sort array by name or score
bool help_sort(Movie *&array, string select, int j)
{
    if (select == "name")
    {
        if (array[j].get_movie_name() > array[j + 1].get_movie_name())
        {
            return true;
        }
        return false;
    }
    else if (select == "score")
    {
        if (array[j].get_score() < array[j + 1].get_score())
        {
```

```
            return true;
        }
        return false;
    }
    else
    {
        throw invalid_argument("ERROR: Movies only can order by name's or score's
!");
    }
    return false;
}
// sorting function
void sort_by(Movie *&array, string select, int &n, bool (*function)(Movie *&, str
ing, int))
{
    bool check = true /* for optimizing bubble sort */, sort = false;
    if (n > 0)
    {
        try
        {
            for (size_t i = 0; i < n && check; i++)
            {
                check = false;
                for (size_t j = 0; j < n - i - 1; j++)
                {
                    if (function(array, select, j))
                    {
                        Movie temp = array[j];
                        array[j] = array[j + 1];
                        array[j + 1] = temp;
                        check = true;
                        sort = true;
                    }
                }
            }
        }
        catch (invalid_argument &errorselected)
        {
            cerr << "ERROR: " << errorselected.what() << " you entered \"" << sel
ect << "\"!" << endl;
        }
    }
    else
    {
        cerr << "--> ERROR: " << endl
             << "No exist any movies for sorting!" << endl;
    }
    if (sort)
```

```
    {
        cout << "--> Movies sorted by " << select << endl;
    }
}
// ----------------------------

// average-score function
// ----------------------------
void avg_score(Movie *array, int n)
{
    if (n > 0)
    {
        float avg = 0;
        for (size_t i = 0; i <= n; i++)
        {
            avg += array[i].get_score();
        }
        float result = avg / n;
        cout << "Average Score: " << result << endl;
    }
    else
    {
        throw invalid_argument("No moives added!");
    }
}
// ----------------------------

// help function
// ----------------------------
void help()
{
    cout << endl
         << "----------------------------------" << endl;
    cout << "Add Movie: "
         << "add < movie name > < score > < production Year > < directory name >
< time > < production country > " << endl;
    cout << "--> Gauide:" << endl
         << "The name of movie be must included letters and numbers" << endl
         << "Score number be must between 0 and 10" << endl
         << "History of production of first movie was 1895 and now we are in 2021
. So you must enter a year between 1895 and 2021" << endl
         << "Time of movie be must longer than 0 min" << endl
         << "The name of director of movie be must included letters, you entered
invalid character's!" << endl
         << "The name of country's production of movie be must included letters"
         << "--> Example : add SpiderMan3 10 2014 nolan 120 Iran"
         << endl
         << endl;
```

```
    cout << "Remove Movie: "
         << "remove < movie name >" << endl
         << "--> Example: remove SpiderMan3"
         << endl
         << endl;

    cout << "Sort Movie's: "
         << "sort_by < score/name >" << endl;
    cout << "--> Gauide:" << endl
         << "Movies can sorted by \" name \" or \" score \""
         << "--> Example: sort_by name"
         << endl
         << endl;

    cout << "Show Movie: "
         << "show < movie name >" << endl
         << "--> Gauide:" << endl
         << "showing data of a movie in a line"
         << "--> Example: show SpiderMan3"
         << endl
         << endl;

    cout << "Show All Movie's: "
         << "show-all" << endl
         << "--> Gauide:" << endl
         << "showing data's of all movie with added !" << endl
         << "--> Example:  show-all"
         << "--> be attention to - in order of show-all"
         << endl
         << endl;

    cout << "Average Score Of Movie's: "
         << "average-score" << endl
         << "--> Gauide:" << endl
         << "calculates average score of all of movies with added" << endl
         << "--> Example: average-score "
         << endl
         << endl;

    cout << "Score Details Of Movie's: "
         << "score-details" << endl
         << "--> Gauide:" << endl
         << "score details get two argument. get and set" << endl
         << "set getting two number. first number is number of reviewer and secon
d number is score's of them!" << endl
         << "and next enter score's" << endl
```

```
         << "get getting two number. first number is number of reviewer and secon
d number is score's number" << endl
         << "--> Example:" << endl
         << "score-details spIderMan3 set 3 2 100 85 80 73 60 70" << endl
         << "score-details SpiderMan3 get 2 2 \t"
         << " print --> 73" << endl;

    cout << "Help: "
         << "help" << endl
         << "--> Gauide:" << endl
         << "showing this list"
         << endl
         << endl;
    cout << "-----------------------------------" << endl;
}
// ----------------------------

// checking movie's list for available movie's
// ----------------------------
bool checkmovie(string temp, Movie *array, int n, int &i /* in show function we u
se i for showing data */)
{
    for (i = 0; i < n; i++)
    {
        if (temp.length() == array[i].get_movie_name().length()) // if name's wer
e similar length of them is equal
        {
            string tempname = temp;
            string namemovie = array[i].get_movie_name();
            for (size_t q = 0; tempname[q] != '\0'; q++)
            {
                if (isupper(tempname[q]))
                {
                    tempname[q] = tolower(tempname[q]);
                }
                if (isupper(namemovie[q]))
                {
                    namemovie[q] = tolower(namemovie[q]);
                }
            }
            if (tempname == namemovie)
            {
                return true;
            }
        }
    }
    return false;
}
```

```
// ----------------------------

// score details function: add many score for a movie
// ----------------------------
void score_details(string tempname, Movie *array, int &n)
{
    int i = 0;
    if (!checkmovie(tempname, array, n, i))
    {
        throw invalid_argument("ERROR: The name with you entered is not available
 in list of movie's!, please enter a valid name");
    }
    string select_score;
    cin >> select_score;
    order_adjust(select_score);
    if (select_score == "set")
    {
        array[i].set_score_details();
        cout << "Score's added for " << array[i].get_movie_name() << endl;
    }
    else if (select_score == "get")
    {
        cout << "Score 2 & 2 for " << array[i].get_movie_name() << " --> ";
        array[i].get_score_details();
    }
    else
    {
        throw invalid_argument("ERROR: You enter invalid argument. Score-
Details can get two value, get and set!");
    }
}
// ----------------------------

// adjusting order (lower and upper case)

void order_adjust(string &select)
{
    // for no sensitive select to lower or upper case
    for (size_t q = 0; select[q] != '\0'; q++)
    {
        if (isupper(select[q]))
        {
            select[q] = tolower(select[q]);
        }
    }
}
// navigator and separator of program
// ----------------------------
```

```
void Command_Separator(string select, Movie *&array, int &n)
{
    order_adjust(select);

    if (select == "add")
    {
        Movie film;
        string tempname, directortemp, countrytemp;
        unsigned int temptime, tempyear;
        float tempscore;
        cin >> tempname >> tempscore >> tempyear >> directortemp >> temptime >> c
ountrytemp;
        film.set_movie_name(tempname);
        film.set_score(tempscore);
        film.set_director(directortemp);
        film.set_year(tempyear);
        film.set_country(countrytemp);
        film.set_random_id();
        // Movie film(tempname, tempscore, tempyear, directortemp, temptime, coun
trytemp);

        // cout << tempname << "\t" << tempscore << "\t" << directortemp << "\t"
<< tempyear << "\t" << countrytemp << endl;
        try
        {
            add(film, array, n);
        }
        catch (invalid_argument &nameofmovie)
        {
            cerr << "ERROR: " << nameofmovie.what() << endl;
        }
    }
    else if (select == "remove")
    {
        string tempname;
        cin >> tempname;
        try
        {
            remove(tempname, array, n);
            cout << tempname << " Removed!" << endl;
        }
        catch (invalid_argument &remove)
        {
            cerr << remove.what() << endl;
        }
    }
    else if (select == "sort-by")
    {
```

```
        string select;
        cin >> select;
        sort_by(array, select, n, &help_sort);
    }
    else if (select == "show")
    {
        string tempname;
        cin >> tempname;
        show(tempname, array, n);
    }
    else if (select == "show-all")
    {
        show_all(array, n);
    }
    else if (select == "average-score")
    {
        try
        {
            avg_score(array, n);
        }
        catch (invalid_argument &avg)
        {
            cerr << "ERROR: " << avg.what() << endl;
        }
    }
    else if (select == "help")
    {
        help();
    }
    else if (select == "score-details")
    {
        string tempname;
        cin >> tempname;
        score_details(tempname, array, n);
    }
    else
    {
        cerr << "ERROR input data" << endl
             << "You Enter: " << select << endl
             << "Plsase Enter \" help \" to show manuals of program! " << endl;
    }
}
// ----------------------------
```

این برنامه را با تعداد کد های کمتر هم قابل پیاده سازی بود اما سعی شده که برنامه ماژولار باشد و به گونه ای طراحی شود که پشتیبانی از آن به راحتی ممکن باشد.

منابع و مراجع

Stackoverflow

C++ How To Program

استاد صنعتی