

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”
КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КУРСОВА РОБОТА
з дисципліни: «Об’єктно-орієнтоване програмування»

На тему:
«База фільмів»

Студента ПЗ-21

спеціальності 6.121

«Інженерія програмного забезпечення»

Бабіля О.О.

Керівник: доцент кафедри ПЗ,

к.т.н., доцент Коротєєва Т. О.

Національна шкала _____

Кількість балів: ____ Оцінка ECTS ____

Члени комісії _____

1. Технічне завдання

1. Розробити програму засобами ООП на мові C++ згідно вказаного варіанту (Варіант №1).
2. Передбачити віконний режим роботи програми та інтерфейс користувача.
3. Передбачити ввід даних у двох режимах:
 - З клавіатури;
 - З файлу;
4. Передбачити у програмі не менше 3-х виняткових ситуацій.
5. Продемонструвати викладачу роботу розробленої програми.
6. Сформуванати звіт курсової роботи обсягом не менше 20 сторінок.

Завдання варіанту №1

Створити таблицю у візуальному середовищі

Назва фільму	Режисер	Рік випуску	Актори	Бюджет	Країна виробництва	Тривалість
--------------	---------	-------------	--------	--------	--------------------	------------

- 1) Алгоритмом простої вибірки відсортувати записи за Країною виробництва.
- 2) Визначити Назви фільмів, в яких однакові режисери та найменші бюджети одночасно.
- 3) За заданою країною виробництва знайти всі фільми, в яких найбільші бюджети і найменша тривалість одночасно.
- 4) Для кожного Режисера визначити фільм з найбільшою тривалістю.
- 5) Знайти найдорожчий та найстарший фільм одночасно.
- 6) За заданим актором визначити всі фільми, в яких він (вона) знімались.
- 7) Встановити найбільш популярного актора.

Для класу створити: 1) Конструктор за замовчуванням; 2) Конструктор з параметрами; 3) конструктор копій; 4) перевизначити операції >>, << для зчитування та запису у файл. Для демонстрації роботи програми використати засоби візуального середовища програмування.

№з/п	Зміст завдання	Дата
1	Здійснити аналітичний огляд літератури за заданою темою та обґрунтувати вибір інструментальних засобів реалізації.	14.10
2	Побудова UML діаграм	15.10
3	Розробка алгоритмів реалізації	16.10
4	Реалізація завдання (кодування)	22.10
5	Формування інструкції користувача	24.10
6	<p>Оформлення звіту до курсової роботи згідно з вимогами Міжнародних стандартів, дотримуючись такої структури:</p> <ul style="list-style-type: none"> · зміст; · алгоритм розв'язку задачі у покроковому представленні; · діаграми UML класів, прецедентів, послідовності виконання; · код розробленої програми з коментарями; · протокол роботи програми для кожного пункту завдання · інструкція користувача та системні вимоги; · опис виняткових ситуацій; · структура файлу вхідних даних; · висновки; · список використаних джерел. 	25.10

Завдання прийнято до виконання: Бабіля (Бабіля О.О.) 22.08.2022

Керівник роботи: _____/Коротєєва Т.О./

Зміст

КУРСОВА РОБОТА	1
1. Технічне завдання	2
2. Алгоритми розв'язку задач	5
3. Діаграми	8
3.1 Діаграма класів	9
3.2 Діаграма прецедентів	9
3.3 Діаграма послідовності	10
4. Код програми.....	11
4.1 Movie.h.....	11
4.2 Movie.cpp	11
4.3 MovieList.h	15
4.4 MovieList.cpp.....	16
4.5 MainWindow.h.....	22
4.6 MainWindow.cpp	23
4.7 main.cpp	27
5. Протокол роботи.....	28
6. Інструкція користувача	35
7. Виняткові ситуації	38
8. Структура файлу вхідних даних	40
Висновки	41
Список використаної літератури	42

2. Алгоритм розв'язку задачі

Для розв'язку задач використовують алгоритми розв'язку [1, 2]

2.1. Задача зчитування з файлу.

Алгоритм RF.

RF 1. Відкриття файлу.

RF 2. Виклик функції зчитування.

RF 3. Перевірка відкриття файлу. Якщо файл не відкрився перехід до пункту RF 8.

RF 4. Перевірка чи файл порожній. Якщо порожній перехід до пункту RF 8.

RF 5. Зчитування даних.

RF 6. Перевірка вмісту файлу. Якщо данні у файлі записані не у відповідному форматі перехід до пункту RF 8.

RF 7. Кінець, успішне зчитування даних. Вихід.

RF 8. Кінець, негативне зчитування даних, повідомлення про помилку. Вихід.

2.2. Задача запису у файл.

Алгоритм WF.

WF1 Відкриття файлу.

WF2. Виклик функції запису.

WF3. Перевірка чи файл відкритий. Якщо файл не відкрився перехід до пункту D5.

WF4. Перевірка чи список фільмів порожній. Якщо так перехід до пункту D5.

WF5. Запис даних.

WF6. Закриття файлу.

WF7. Кінець, успішне запис даних. Вихід.

WF8. Кінець, негативне запис даних, повідомлення про помилку. Вихід.

2.3. Задача сортування списку методом вибірки за країною виробництва.

Алгоритм SS.

R – вхідний масив, i, j – індекси, n – розмір масиву, \min – мінімальне значення, R_0 – перший елемент.

SS1. Виклик функції для сортування.

SS2. Перевірка чи список фільмів порожній. Якщо так перехід до пункту SS8.

SS3. Цикл за індексом проходження. Повторювати кроки SS4 – SS6, при $i=1..n-1$

SS4. Зафіксувати перший поточний елемент: встановити $R_0 = R_i$

SS5. Пошук найменшого значення $\min R_j$ для елементів з індексом $j=i+1, i+2, \dots, n$.

SS6. Перестановка елементів. Якщо $\min R_j < R_0$ та $j \neq i$, то $\min = R_j \leftrightarrow R_0$.

SS7. Кінець, успішне сортування. Вихід.

SS8. Кінець, негативне сортування даних, повідомлення про помилку. Вихід.

2.4. Задача знаходження фільмів з однаковими режисерами та найменшим бюджетом.

Алгоритм SDLB.

SDLB 1. Виклик функції пошуку.

SDLB 2. Перевірка чи список фільмів порожній. Якщо так перехід до пункту SDLB 6.

SDLB 3. Створюємо допоміжний список фільмів та сортуємо його за бюджетом.

SDLB 4. Перевірка чи існує більше 1 фільму з однаковим режисером та найменшим бюджетом.

SDLB 5. Кінець, успішне знаходження фільмів. Вихід.

SDLB 6. Кінець, негативне знаходження фільмів, повідомлення про помилку. Вихід.

2.5. Задача знаходження фільмів з найбільшим бюджетом та найменшою тривалістю за заданою країною виробництва.

Алгоритм FCBB.

FCBB 1. Виклик функції пошуку.

FCBB 2. Перевірка чи список фільмів порожній. Якщо так перехід до пункту FCBB 10.

FCBB 3. Перевірка чи країна задана правильно . Якщо ні перехід до пункту FCBB 10.

FCBB 4. Створюємо два допоміжні списки фільмів та записуємо в них фільми з заданою країною виробництва

FCBB 5. Перевірка чи існують фільми з заданою країною. Якщо ні перехід до пункту FCBB 10.

FCBB 6. Сортуюмо перший список за бюджетом, другий за довжиною.

FCBB 7. Перевірка чи існують які задовільняють умову . Якщо ні перехід до пункту FCBB 10.

FCBB 8. Створення стрічки з назвами фільмів, які задовільняють умову.

FCBB 9. Кінець, успішне знаходження фільмів. Вихід.

FCBB 10. Кінець, негативне знаходження фільмів, повідомлення про помилку. Вихід.

2.6. Задача знаходження фільмів для кожного режисера з найбільшою тривалістю.

Алгоритм LMPD.

LMPD 1. Виклик функції пошуку.

LMPD 2. Перевірка чи список фільмів порожній. Якщо так перехід до пункту LMPD 9.

LMPD 3. Створюємо допоміжний список фільмів та сортуємо його за за тривалістю.

LMPD 4. Відокремлення фільмів якщо є ще фільми від даного режисера.

LMPD 5. Запис даних, якщо режисер має лише 1 фільм.

LMPD 6. Знаходження найдовшого фільму для режисера якщо він має більше 1 фільму.

LMPD 7. Запис даних, якщо режисер має більше 1 фільм.

LMPD 8. Кінець, успішне знаходження фільмів. Вихід.

LMPD 9. Кінець, негативне знаходження фільмів, повідомлення про помилку. Вихід. помилку. Вихід.

2.7. Задача знаходження найстарішого та найдорожчого фільму.

Алгоритм OMEM.

OMEM 1. Виклик функції пошуку.

OMEM 2. Перевірка чи список фільмів порожній. Якщо так перехід до пункту OMEM 6.

OMEM 3. Створюємо два допоміжні списки фільмів та сортуємо перший за бюджетом, другий за роком випуску.

OMEM 4. Пошук фільму одночасно найстарішого та найдорожчого фільму. Якщо такого нема перехід до пункту OMEM 6.

OMEM 5. Кінець, успішне знаходження фільмів. Вихід.

OMEM 6. Кінець, негативне знаходження фільмів, повідомлення про помилку. Вихід.

2.8. Задача знаходження фільмів з де знімався заданий актор.

Алгоритм АМВА.

АМВА 1. Виклик функції пошуку.

АМВА 2. Перевірка чи список фільмів порожній. Якщо так перехід до пункту АМВА 10.

АМВА 3. Перевірка чи актор заданий правильно . Якщо ні перехід до пункту АМВА 8.

АМВА 4. Шукаєм фільми з заданим актором, зберігаєм ці фільми

АМВА 5. Перевірка чи існують фільми з заданим актором. Якщо ні перехід до пункту АМВА 8.

АМВА 6. Створення стрічки з назвами фільмів, які задовільняють умову.

АМВА 7. Кінець, успішне знаходження фільмів. Вихід.

АМВА 8. Кінець, негативне знаходження фільмів, повідомлення про помилку. Вихід.

2.9. Задача знаходження найбільш популярного актору.

Алгоритм МРА.

МРА 1. Виклик функції пошуку.

МРА 2. Перевірка чи список фільмів порожній. Якщо так перехід до пункту МРА 9.

МРА 3. Створюємо стрічку з всіма акторами з списку.

МРА 4. Ділимо цю стрічку на підстрічки

МРА 5. Знаходимо актора який трапляється найбільше.

МРА 6. Створення стрічки з назвами фільмів, які задовільняють умову.

МРА 7. Перевірка чи існує такий актор. Якщо ні перехід до пункту МРА 9.

МРА 8. Кінець, успішне знаходження актору. Вихід.

МРА 9. Кінець, негативне знаходження актору, повідомлення про помилку. Вихід.

3. Діаграми

3.1 Діаграма класів

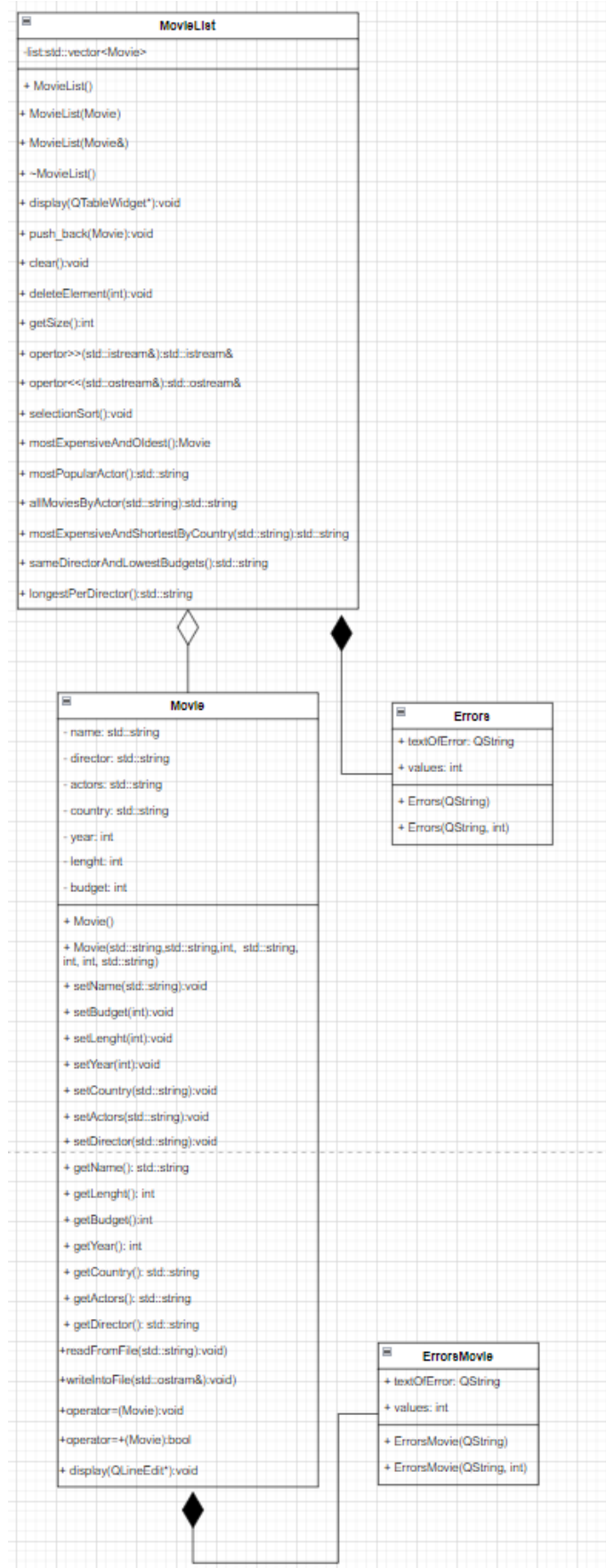


Рис.3.1. UML-діаграма класів

3.2 Діаграма прецедентів

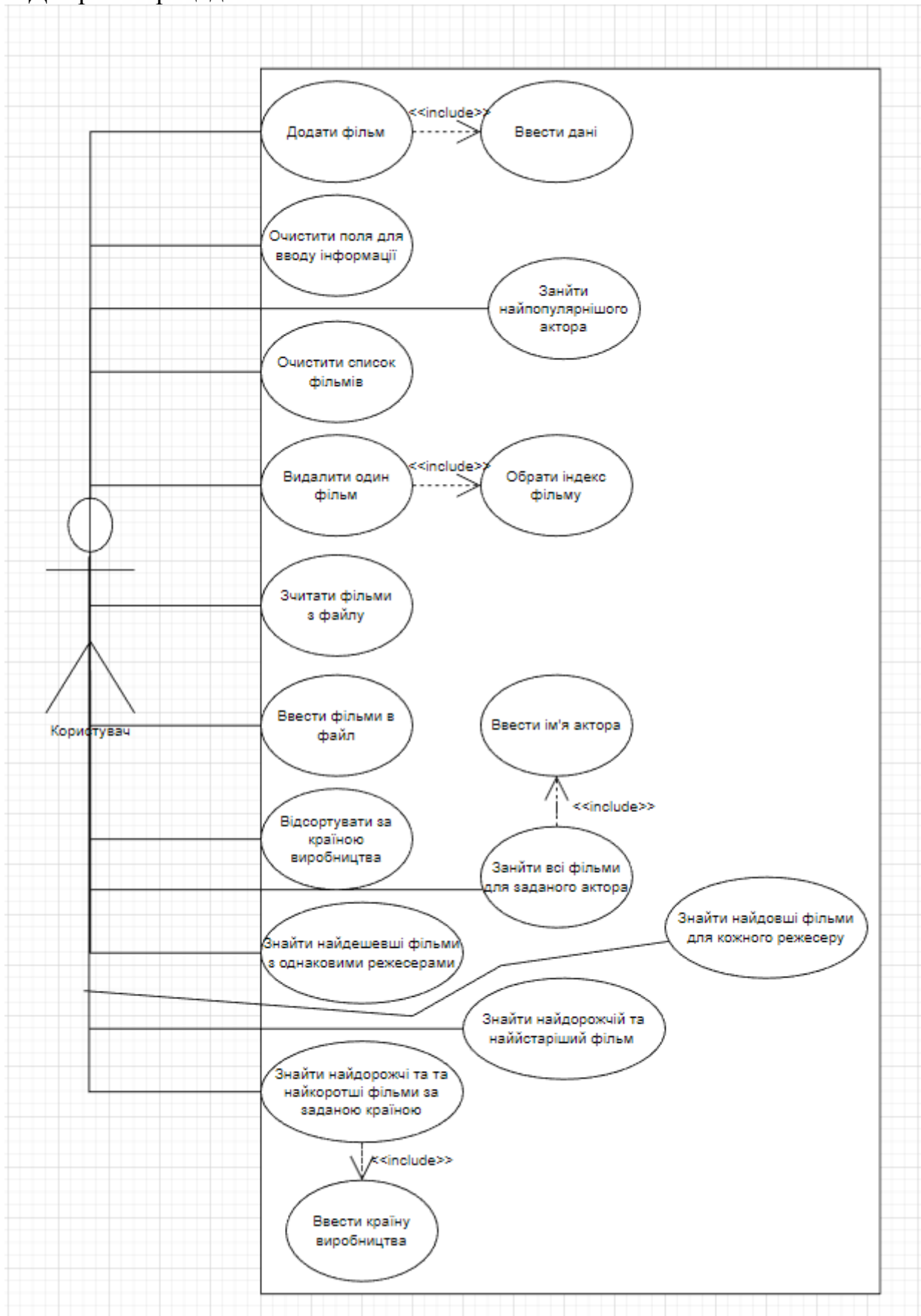


Рис.3.2. UML-діаграма прецедентів

3.3 Діаграма послідовності

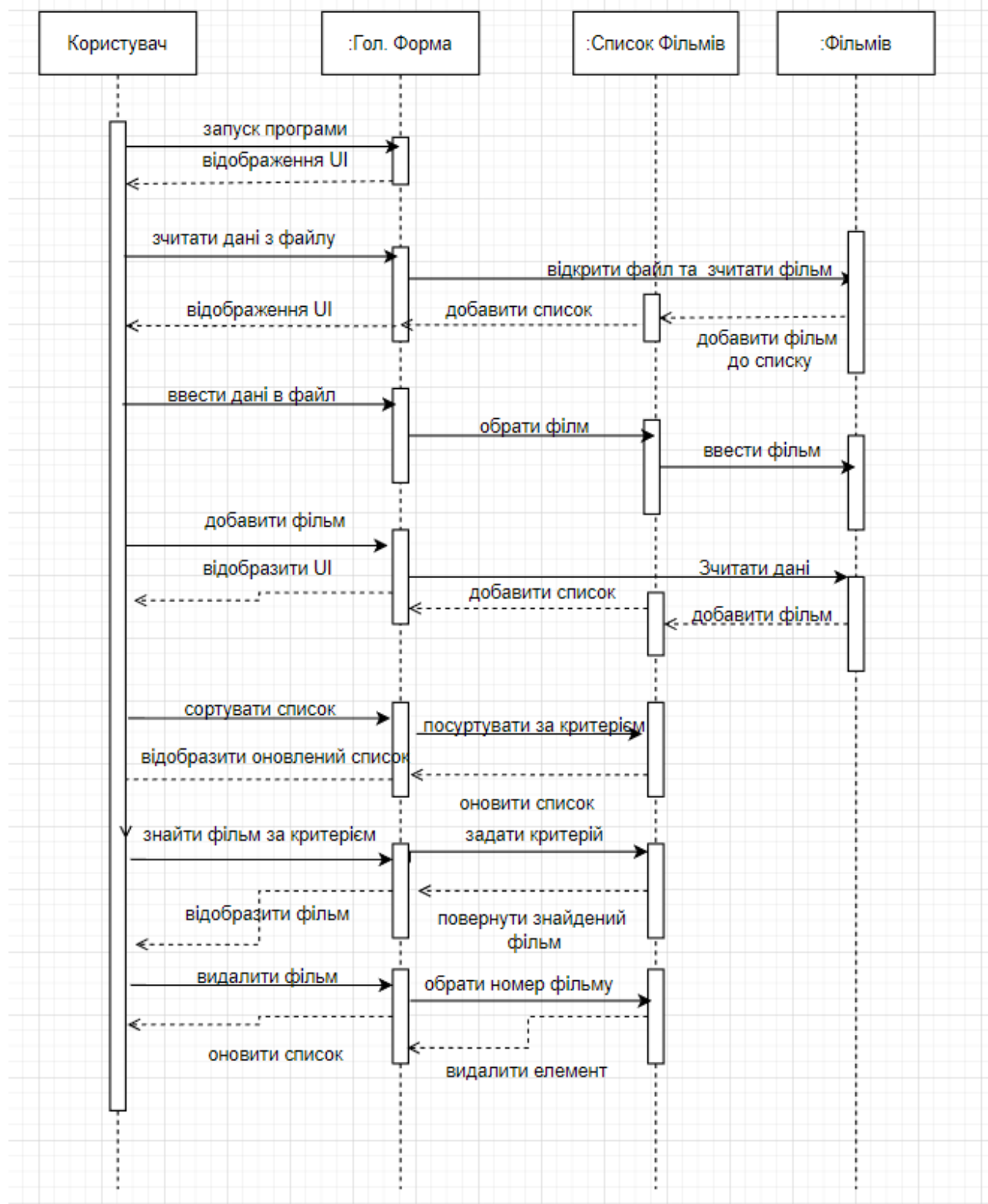


Рис.3.3. UML-діаграма послідовності

4. Код програми

4.1 Movie.h

```
#ifndef MOVIE_H
#define MOVIE_H

#include <iostream>
#include <ostream>
#include <fstream>
#include <string>
#include <vector>
#include <algorithm>
#include <numeric>
#include <QPlainTextEdit>

class Movie
{
private:
    std::string name;
    std::string director;
    int year;
    std::string actors;
    int budget;
    int lenght;
    std::string country;
public:
    class ErrorsMovie
    {
    public:
        QString textOfError;
        int valuse;
        ErrorsMovie(QString usersErrorText, int usersErrorValues);
        ErrorsMovie(QString usersErrorText);
    };
    Movie();
    Movie(std::string, std::string, int, std::string, int, int, std::string);
    std::string getName();
    std::string getDirector();
    int getYear();
    std::string getActors();
    long int getBudget();
    float getLenght();
    std::string getCountry();
    void setName(std::string);
    void setDirector(std::string);
    void setYear(int);
    void setActors(std::string);
    void setBudget(long int);
    void setLenght(float);
    void setCountry(std::string);
    void readFromFile(std::string);
    void writeIntoFile(std::ostream&);
    void operator = (Movie);
    bool operator==(Movie);
    void display(QPlainTextEdit*);
};

#endif // MOVIE_H
```

4.2 Movie.cpp

```
#include "movie.h"
```

```

//Конструктор за замовчуванням
Movie::Movie()
{
    name = "";
    director = "";
    year = 0;
    actors = "";
    budget = 0;
    lenght = 0;
    country = "";
}

//Конструктор з параметрами
Movie::Movie(std::string usersName, std::string usersDirector, int usersYear,
std::string usersActors
, int usersBudget, int usersLength, std::string usersCountry)
{
    //Перевірка даних
    if( usersName=="|| usersDirector=="|| usersYear < 0 || usersActors=="
|| usersBudget<0|| usersLength<0|| usersCountry==" ) throw
ErrorsMovie("Error, checl input data");
    name = usersName;
    director = usersDirector;
    year = usersYear;
    actors = usersActors;
    budget = usersBudget;
    lenght = usersLength;
    country = usersCountry;
}

//Конструктор виняткових с ситуація з 2 параметрами
Movie::ErrorsMovie::ErrorsMovie(QString usersErrorText, int usersErrorValues)
{
    textOfError = usersErrorText;
    valuse = usersErrorValues;
}

//Конструктор виняткових с ситуація з 1 параметром
Movie::ErrorsMovie::ErrorsMovie(QString usersErrorText)
{
    textOfError = usersErrorText;
}

//Геттери для полів
std::string Movie::getName()
{
    return name;
}
std::string Movie::getDirector()
{
    return director;
}
int Movie::getYear()
{
    return year;
}
std::string Movie::getActors()
{
    return actors;
}
long int Movie::getBudget()
{

```

```

        return budget;
    }
    float Movie::getLenght()
    {
        return lenght;
    }

    std::string Movie::getCountry()
    {
        return country;
    }

    //Сеттери для полів
    void Movie::setName(std::string usersName)
    {
        name = usersName;
    }
    void Movie::setDirector(std::string usersDirector)
    {
        director = usersDirector;
    }

    void Movie::setYear(int usersYear)
    {
        year = usersYear;
    }

    void Movie::setActors(std::string usersActors)
    {
        actors = usersActors;
    }

    void Movie::setBudget(long int usersBudget)
    {
        budget = usersBudget;
    }

    void Movie::setLenght(float usersLength)
    {
        lenght = usersLength;
    }

    void Movie::setCountry(std::string usersCountry)
    {
        country = usersCountry;
    }

    //Вивід фільму
    void Movie::display(QPlainTextEdit* pl)
    {
        QString str = QString::fromStdString("Name: " + name + " \nDirector: " +
        director + "\nYear: " + std::to_string(year) + " \nActors: "
        + actors + "\nBudget: " + std::to_string(budget) + " mil$\nLenght: " +
        std::to_string(lenght) + " min\nCountry: " + country);
        pl->setPlainText(str);
    }

    //Зчитування фільма з файлу
    void Movie::readFromFile( std::string str)
    {
        //Знаходимо назву фільма
        int found1 = str.find("\"");
        int found2 = str.find("\"", found1+1);
    }

```

```

//Переві чи назва фільма була знайдена
if(found1==std::string::npos||found2==std::string::npos) throw
ErrorsMovie("Error, checl input data");
name = str.substr(found1+1, found2-3);
//Знаходимо режисера фільма
found1 = str.find("'", found2+1);
found2 = str.find("'", found1+1);
//Переві чи режисер фільма була знайдена
if(found1==std::string::npos||found2==std::string::npos) throw
ErrorsMovie("Error, checl input data");
director = str.substr(found1+1, found2-name.size()-6);
//Знаходимо рік випуску фільма
found1 = str.find("'", found2+1);
found2 = str.find("'", found1+1);
//Переві чи рік випуску фільма був знайдена
if(found1==std::string::npos||found2==std::string::npos) throw
ErrorsMovie("Error, checl input data");
year = stoi(str.substr(found1 + 1, found2 - 1));
//Знаходимо акторів фільма
found1 = str.find("'", found2 + 1);
found2 = str.find("'", found1 + 1);
//Переві чи актори фільма були знайдена
if(found1==std::string::npos||found2==std::string::npos) throw
ErrorsMovie("Error, checl input data");
actors = str.substr(found1 + 1, found2 - name.size() - director.size()-16);
//Знаходимо бюджет фільма
found1 = str.find("'", found2 + 1);
found2 = str.find("'", found1 + 1);
//Переві чи бюджет фільма було знайдена
if(found1==std::string::npos||found2==std::string::npos) throw
ErrorsMovie("Error, checl input data");
budget = stoi(str.substr(found1 + 1, found2 - name.size() - director.size()-
actors.size() - 18));
//Знаходимо тривалість фільма
found1 = str.find("'", found2 + 1);
found2 = str.find("'", found1 + 1);
//Переві чи тривалість фільма була знайдена
if(found1==std::string::npos||found2==std::string::npos) throw
ErrorsMovie("Error, checl input data");
lenght = stoi(str.substr(found1 + 1, found2 - name.size() - director.size()-
actors.size() - 24));
//Знаходимо країну виробництва фільма
found1 = str.find("'", found2 + 1);
//Переві чи країну виробництва фільма була знайдена
if(found1==std::string::npos||found2==std::string::npos) throw
ErrorsMovie("Error, checl input data");
country = str.substr(found1 + 1, 3 );
}
//Ввід фільмів до файлу
void Movie::writeIntoFile(std::ostream& file)
{
    file << name << ' ' << director << ' ' << year << ' ' << actors << ' ' <<
budget << ' ' << lenght << ' ' << country<<std::endl; //<< director << year
}

//Оператор присвоєння
void Movie::operator = (Movie a)
{
    name = a.name;
    director = a.director;
    year = a.year;
    actors = a.actors;
}

```

```

    budget = a.budget;
    lenght = a.lenght;
    country = a.country;
}

//Оператор порівняння
bool Movie::operator==(Movie a)
{
    return
        name == a.name &&
        director == a.director &&
        year == a.year &&
        actors == a.actors &&
        budget == a.budget &&
        lenght == a.lenght &&
        country == a.country;
}

```

4.3 MovieList.h

```

#ifndef MOVIELIST_H
#define MOVIELIST_H

#include <iostream>
#include <ostream>
#include <fstream>
#include <string>
#include <vector>
#include <algorithm>
#include <numeric>
#include "movie.h"
#include <QTableWidget>

class MovieList
{
private:
    std::vector<Movie> list;
public:
    class Errors
    {
    public:
        QString textOfError;
        int valuse;
        Errors(QString usersErrorText, int usersErrorValues);
        Errors(QString usersErrorText);
    };
    MovieList();
    MovieList(Movie);
    MovieList(MovieList&);
    ~MovieList();
    void display(QTableWidget*);
    void push_back(Movie);
    void clear();
    void deleteElement(int);
    int getSize();
    std::istream& operator >> (std::istream&);
    std::ostream& operator << (std::ostream&);
    void selectionSort();
    Movie mostExpensiveAndOldest();
    std::string mostPopularActor();
    std::string allMoviesByActor(std::string);
    std::string mostExpensiveAndShortestByCountry(std::string);
    std::string sameDirectorsAndLowestBudgets();
}

```

```

        std::string longestPerDirector();
};
#endif // MOVIELIST_H

```

4.4 MovieList.cpp

```

#include "movielist.h"

//Конструктор за замовчуванням
MovieList::MovieList()
{
    list.clear();
}

//Конструктор з параметром
MovieList::MovieList(Movie a)
{
    list.push_back(a);
}

//Конструктор копіювання
MovieList::MovieList(MovieList& a)
{
    list = a.list;
}

//Деструктор
MovieList::~MovieList()
{
    list.clear();
}

//Конструктор виняткових ситуацій з 2 параметрами
MovieList::Errors::Errors(QString usersErrorText, int usersErrorValues)
{
    textOfError = usersErrorText;
    valuse = usersErrorValues;
}

//Конструктор виняткових ситуацій з 1 параметрами
MovieList::Errors::Errors(QString usersErrorText)
{
    textOfError = usersErrorText;
}

//Вивід даних до списку
void MovieList::display(QTableWidget* tw)
{
    tw->setRowCount(list.size());
    for (int i = 0; i < list.size(); i++)
    {
        QTableWidgetItem *name = new
        QTableWidgetItem(QString::fromStdString(list[i].getName()));
        tw->setItem(i, 0, name);
        QTableWidgetItem *director = new
        QTableWidgetItem(QString::fromStdString(list[i].getDirector()));
        tw->setItem(i, 1, director);
        QTableWidgetItem *year = new
        QTableWidgetItem(QString::number(list[i].getYear()));
    }
}

```



```

        tw->setItem(i, 2, year);
        QTableWidgetItem *actors = new
QTableWidgetItem(QString::fromStdString(list[i].getActors()));
        tw->setItem(i, 3, actors);
        QTableWidgetItem *budget = new
QTableWidgetItem(QString::number(list[i].getBudget()));
        tw->setItem(i, 4, budget);
        QTableWidgetItem *length = new
QTableWidgetItem(QString::number(list[i].getLenght()));
        tw->setItem(i, 6, length);
        QTableWidgetItem *country = new
QTableWidgetItem(QString::fromStdString(list[i].getCountry()));
        tw->setItem(i, 5, country);
    }
}

//Зчитування фільмів з файду
std::istream& MovieList::operator >> (std::istream& infile)
{
    //Перевірка відкриття файлу
    if(!infile) throw Errors("Error, file cannot be found");
    //Перевірка чи файл порожній
    if(infile.peek() == std::ifstream::traits_type::eof()) throw Errors("Error,
file is empty");

    Movie tmp;
    std::string str;
    try
    {
        //Пострічкове зчитування з файлу
        while (std::getline(infile, str))
        {
            tmp.readFromFile(str);
            list.push_back(tmp);
        }
    }
    catch (Movie::ErrorsMovie& error)
    {
        throw Errors(error.textOfError);
    }

    return infile;
}

//Запис фільмів до файду
std::ostream& MovieList::operator << (std::ostream& outfile)
{
    //Перевірка відкриття файлу
    if(!outfile) throw Errors("Error, file cannot be found");
    //Перевірка чи список фільмів порожній
    if(list.size() == 0) throw Errors("Error, list is empty");
    //Пострічковий ввід фільмів до файлу
    for (int i = 0; i < list.size(); i++)
        list[i].writeIntoFile(outfile);
    return outfile;
}

//Додання одного фільму
void MovieList::push_back(Movie a)
{
    list.push_back(a);
}

```

```

//Очищення списку
void MovieList::clear()
{
    list.clear();
}

//Видалення одного елементу
void MovieList::deleteElement(int index)
{
    if(list.size()==0) throw Errors("Error, list is empty");
    list.erase(list.begin() + index);
}

//Розмір масиву
int MovieList::getSize()
{
    return list.size();
}

//Сортування вибіркою за країною
void MovieList::selectionSort()
{
    //Перевірка чи файлу порожній
    if(list.size()==0) throw Errors("Error, list is empty");

    int i, j, min_idx;

    for (i = 0; i < list.size() - 1; i++)
    {
        min_idx = i;
        for (j = i + 1; j < list.size(); j++)
            if (list[j].getCountry() < list[min_idx].getCountry())
                min_idx = j;

        if (min_idx != i)
            std::swap(list[i], list[min_idx]);
    }
}

//Знаходження найстарішого та найдорожчого фільму.
Movie MovieList::mostExpensiveAndOldest()
{
    if(list.size()==0) throw Errors("Error, list is empty");
    //Створюємо два допоміжні списки фільмів та сортуємо перший за бюджетом,
    //другий за роком випуску.
    std::vector<Movie> listForBudget = list;
    std::vector<Movie> listForYear = list;

    std::sort(listForBudget.begin(), listForBudget.end(),
        [](Movie a, Movie b) {
            return a.getBudget() > b.getBudget();
        });
    std::sort(listForYear.begin(), listForYear.end(),
        [](Movie a, Movie b) {
            return a.getYear() > b.getYear();
        });
    //Пошук фільму одночасно найстарішого та найдорожчого фільму.
    for (int i = 0; i < list.size(); i++)
        if (listForBudget[i] == listForYear[i]) return listForBudget[i];

    throw(Errors("Warning, no such movie"));
}

```

```

}

//Знаходження найбільш популярного актору.
std::string MovieList::mostPopularActor()
{
    if(list.size()==0) throw Errors("Error, list is empty");
    //Створюємо стрічку з всіма акторами з списку
    std::string str;
    for (Movie el : list)
        str += el.getActors()+" ";
    std::vector<std::string> stringVec;

    int found1 , found2, found3=0, sum = 1, i = 0;
    //Ділимо цю стрічку на підстрічки
    while (str.find(" ", found3+1))
    {
        found1 = found3++;
        found2 = str.find(" ", ++found1);
        found3 = str.find(" ", ++found2);
        stringVec.push_back(str.substr(found1, found3-sum));
        sum += stringVec[i++].size() + 1;
    }

    int arrSize = stringVec.size();
    for (int k = 4; k < stringVec.size(); k += 4)
        stringVec.erase(std::next(stringVec.begin(), k));
    //Знаходимо актора який лягає найбільше.
    int index, max=0;
    for (int k = 0; k < stringVec.size(); k++)
    {
        if (max < std::count(stringVec.begin(), stringVec.end(), stringVec[k]))
        {
            max = std::count(stringVec.begin(), stringVec.end(), stringVec[k]);
            index = k;
        }
    }
    if(stringVec.empty()) throw(Errors("Warning, no such actor"));
    return stringVec[index];
}

//Знаходження фільмів з де знімався заданий актор.
std::string MovieList::allMoviesByActor(std::string actor)
{
    //Перевірка чи список порожній
    if(list.size()==0) throw Errors("Error, list is empty");
    //Перевірка формату актора
    if( actor=="")
    {
        throw(Errors("Error,check input"));
    }
    std::vector<int> index;
    std::string tmp, listOfMovies;
    //Шукаємо фільми з заданим актором, зберігаємо ці фільми
    for (int i = 0; i < list.size(); i++)
    {
        tmp = list[i].getActors();
        if (tmp.find(actor,0) != std::string::npos) index.push_back(i);
    }

    if(index.empty())
    {
        throw(Errors("Warning, no such movie"));
    }
    for (int i = 0; i < index.size()-1; i++)

```

```

        listOfMovies += list[index[i]].getName() + ", ";

    return listOfMovies + list[index[index.size()-1]].getName();
}

//Знаходження фільмів з найбільшим бюджетом та найменшою тривалістю за заданою
країною виробництва.
std::string MovieList::mostExpensiveAndShortestByCountry(std::string country)
{
    //Перевірка чи список порожній
    if(list.size()==0) throw Errors("Error, list is empty");
    //Перевірка формату країни
    if( country=="")
    {
        throw(Errors("Error, check input data"));
    }

    std::vector<Movie> listForBudget;
    std::vector<Movie> listForLenght;
    //Створюємо два допоміжні списки фільмів та записуємо в них фільми з заданою
    країною виробництва
    for(Movie el:list)
    {
        if(el.getCountry()!=country) continue;;
        listForBudget.push_back(el);
        listForLenght.push_back(el);
    }
    //Перевірка чи існують фільми з заданою країною
    if(listForBudget.empty())
    {
        throw(Errors("No such movies"));
    }

    std::sort(listForBudget.begin(), listForBudget.end(),
        [](Movie a, Movie b) {
            return a.getBudget() < b.getBudget();
        });
    std::sort(listForLenght.begin(), listForLenght.end(),
        [](Movie a, Movie b) {
            return a.getLenght() > b.getLenght();
        });

    std::vector<int> index;

    for (int i = 0; i < listForBudget.size(); i++)
        if ((listForBudget[i].getCountry() == country &&
listForLenght[i].getCountry() == country)&&
            (listForBudget[i].getLenght() == listForLenght[i].getLenght())&&
            (listForBudget[i].getBudget() == listForLenght[i].getBudget()))
            index.push_back(i);
    std::string tmp="";

    if(index.empty())
    {
        throw(Errors("No such movies"));
    }

    for (int i = 0; i < index.size()-1; i++)
        tmp+=listForBudget[index[i]].getName()+" , ";

    return country+" movie with biggest budget and shortest length is " +
tmp+listForBudget[listForBudget.size()-1].getName();
}

```

```

}
//Знаходження фільмів з однаковими режисерами та найменшим бюджетом.
std::string MovieList::sameDirectorsAndLowestBudgets()
{
    //Перевірка чи список порожній
    if(list.size()==0) throw Errors("Error, list is empty");
    //Створюємо допоміжний список та сортуємо його за бюджетом
    std::vector<Movie> listForBudget = list;

    std::sort(listForBudget.begin(), listForBudget.end(),
        [](Movie a, Movie b) {
            return a.getBudget() < b.getBudget();
        });

    std::vector<int> index;

    std::string cmp = listForBudget[0].getDirector();
    index.push_back(0);
    //Перевірка чи існує більше 1 фільму з однаковим режисером та найменшим
    бюджетом.
    for (int i = 1; i < list.size(); i++)
        if (listForBudget[i].getDirector() == cmp)
            index.push_back(i);

    std::string tmp = "";
    for (int i = 0; i < index.size()-1; i++)
        tmp += listForBudget[index[i]].getName() + ", ";

    if(index.size()==1) return "No such movies";
    return tmp+listForBudget[index[index.size()-1]].getName();
}

//Знаходження фільмів для кожного режисера з найбільшою тривалістю
std::string MovieList::longestPerDirector()
{
    if(list.size()==0) throw Errors("Error, list is empty");

    std::vector<Movie> listForDirector = list;
    std::vector<int> index;
    //Створюємо два допоміжний список фільмів та сортуємо його за за тривалістю.
    std::sort(listForDirector.begin(), listForDirector.end(),
        [](Movie a, Movie b) {
            return a.getDirector() > b.getDirector();
        });

    std::string str = "";
    //Відокремлення фільмів якщо є ще фільми від даного режисера.
    for (int i = 0; i < listForDirector.size(); i++)
    {
        std::string tmp = listForDirector[i].getDirector();
        if (std::count_if(listForDirector.begin(), listForDirector.end(),
            [&tmp](Movie a) {
                return a.getDirector() == tmp;
            }) > 1) index.push_back(i);
        else str += "    "+listForDirector[i].getDirector() + " - " +
listForDirector[i].getName() + "\n";
    }
    //Знаходження найдовшого фільму для режисера якщо він має більше 1 фільму.
    int max = 0, indexOfMax = 0, j = 0;;
    std::string name = listForDirector[index[0]].getDirector();
    for (int i = 0; i < index.size(); i++)
    {

```

```

        if (listForDirector[index[i]].getDirector() == name)
        {
            if (max < listForDirector[index[i]].getLenght())
            {
                max = listForDirector[index[i]].getLenght();
                indexOfMax = index[i];
            }
        }
        else
        {
            str += "      "+listForDirector[indexOfMax].getDirector() + " - " +
listForDirector[indexOfMax].getName() + "\n";

            name = listForDirector[index[i+ 1]].getDirector();
            max = 0;
            indexOfMax = 0;
            i--;
        }
    }
    str += "      "+listForDirector[indexOfMax].getDirector() + " - " +
listForDirector[indexOfMax].getName() + "\n";
    std::cout << std::endl;

    return str;
}

```

4.5 MainWindow.h

```

#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include "movie.h"
#include "movielist.h"
#include <QMessageBox>

QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();

private slots:
    void on_readFromFileButton_clicked();

    void on_addMovieButton_clicked();

    void on_clearButton_clicked();

    void on_writeIntoFileButton_clicked();

    void on_sortButton_clicked();

    void on_mostPopularActorButton_clicked();

    void on_allMoviedByActorButton_clicked();

```

```

void on_longestMoviePerDirButton_clicked();

void on_sameDirLowBudButton_clicked();

void on_longesAndMostExpensiveButton_clicked();

void on_mostExpensiveAndShortestButton_clicked();

void on_clearListButton_clicked();

void on_deleteButton_clicked();

private:
    Ui::MainWindow *ui;
};
#endif // MAINWINDOW_H

```

4.6 MainWindow.cpp

```

#include "mainwindow.h"
#include "ui_mainwindow.h"

MovieList list;

MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    //ui->tableWidget->setRowCount(10);
    ui->tableWidget->setColumnWidth(1,125);
    ui->tableWidget->setColumnWidth(2,60);
    ui->tableWidget->setColumnWidth(3,407);
    ui->tableWidget->setColumnWidth(4,100);
    ui->tableWidget->setColumnWidth(6,100);
    ui->tableWidget->setColumnWidth(5,60);
    ui->yearSpinBox->setMinimum(1500);
    ui->yearSpinBox->setMaximum(2022);
}

MainWindow::~MainWindow()
{
    delete ui;
}

//Зчитування з файлу
void MainWindow::on_readFromFileButton_clicked()
{
    try
    {
        std::ifstream infile("C:\\Users\\home\\Desktop\\Movies.txt");
        list >> infile;
        list.display(ui->tableWidget);
        ui->indexSpinBox->setMaximum(list.getSize());
    }
    catch (MovieList::Errors& error)
    {
        QMessageBox::critical(this,"Error",error.textOfError);
    }
}

//Додання одного фільму

```

```

void MainWindow::on_addMovieButton_clicked()
{
    //Перевірки даних
    if(ui->nameLineEdit->text()=="|||
        ui->directorLineEdit->text()=="|||
        ui->yearSpinBox->text()=="0"|||
        ui->actorsLineEdit->text()=="|||
        ui->budgetLineEdit->text().toInt()<0||
        ui->budgetLineEdit->text()=="|||
        ui->lengthLineEdit->text().toInt()<0||
        ui->lengthLineEdit->text()=="|||
        ui->countryLineEdit->text()=="")
    {
        QMessageBox::critical(this, "Error", "Error, check input data");
        return;
    }
    try
    {
        Movie tmp(ui->nameLineEdit->text().toStdString(),
            ui->directorLineEdit->text().toStdString(),
            ui->yearSpinBox->text().toInt(),
            ui->actorsLineEdit->text().toStdString(),
            ui->budgetLineEdit->text().toInt(),
            ui->lengthLineEdit->text().toInt(),
            ui->countryLineEdit->text().toStdString());

        list.push_back(tmp);
        list.display(ui->tableWidget);

        ui->indexSpinBox->setMaximum(list.getSize());
        ui->nameLineEdit->setText("");
        ui->directorLineEdit->setText("");
        ui->yearSpinBox->setValue(0);
        ui->actorsLineEdit->setText("");
        ui->budgetLineEdit->setText("");
        ui->lengthLineEdit->setText("");
        ui->countryLineEdit->setText("");
    }
    catch (Movie::ErrorsMovie& error)
    {
        QMessageBox::critical(this, "Error", error.textOfError());
    }
}

//Очищення полів для вводу інформації
void MainWindow::on_clearButton_clicked()
{
    ui->nameLineEdit->setText("");
    ui->directorLineEdit->setText("");
    ui->yearSpinBox->setValue(0);
    ui->actorsLineEdit->setText("");
    ui->budgetLineEdit->setText("");
    ui->lengthLineEdit->setText("");
    ui->countryLineEdit->setText("");
}

//Запис фільмів у файл
void MainWindow::on_writeIntoFileButton_clicked()
{
    try
    {
        std::ofstream outfile("C:\\Users\\home\\Desktop\\OutPut.txt",
std::ios::app);

```



```

        list << outfile;
    }
    catch (MovieList::Errors& error)
    {
        QMessageBox::critical(this, "Error", error.textOfError);
    }
}

//Сортування вибіркою
void MainWindow::on_sortButton_clicked()
{
    try
    {
        list.selectionSort();
        list.display(ui->tableWidget);
    }
    catch (MovieList::Errors& error)
    {
        QMessageBox::critical(this, "Error", error.textOfError);
    }
}

//Знаходження найбільш популярного актору
void MainWindow::on_mostPopularActorButton_clicked()
{
    try
    {
        QString str = QString::fromStdString(list.mostPopularActor());
        ui->plainTextEdit->setPlainText("Most popular actor is "+str);
    }
    catch (MovieList::Errors& error)
    {
        QMessageBox::warning(this, "Warning", error.textOfError);
    }
}

//Знаходження фільмів з де знімався заданий актор.
void MainWindow::on_allMoviedByActorButton_clicked()
{
    try
    {
        QString str = QString::fromStdString(list.allMoviesByActor(
            ui->searchActorLineEidt->text().toStdString()));
        ui->plainTextEdit->setPlainText(ui->searchActorLineEidt->text()+"
played in "+str);
    }
    catch (MovieList::Errors& error)
    {
        if(error.textOfError=="Error, check
input") QMessageBox::critical(this, "Error", error.textOfError);
        else QMessageBox::warning(this, "Warning", error.textOfError);
    }
}

//Знаходження фільмів для кожного режисера з найбільшою тривалістю
void MainWindow::on_longestMoviePerDirButton_clicked()
{
    try
    {

```

```

        QString str = QString::fromStdString("Longest movies per Director
are:\n\n" + list.longestPerDirector());
        ui->plainTextEdit->setPlainText(str);
    }
    catch (MovieList::Errors& error)
    {
        QMessageBox::critical(this, "Error", error.textOfError);
    }
}

//Знаходження фільмів з однаковими режисерами та найменшим бюджетом.
void MainWindow::on_sameDirLowBudButton_clicked()
{
    try
    {
        QString str = QString::fromStdString("Movies with same Directors and
Lowest budgets are: " + list.sameDirectorsAndLowestBudgets());
        ui->plainTextEdit->setPlainText(str);
    }
    catch (MovieList::Errors& error)
    {
        QMessageBox::critical(this, "Error", error.textOfError);
    }
}

//Знаходження найстарішого та найдорожчого фільму.
void MainWindow::on_longesAndMostExpensiveButton_clicked()
{
    try
    {
        Movie a = list.mostExpensiveAndOldest();
        a.display(ui->plainTextEdit);
    }
    catch (MovieList::Errors& error)
    {
        QMessageBox::warning(this, "Warning", error.textOfError);
    }
    catch (Movie::ErrorsMovie& error)
    {
        QMessageBox::warning(this, "Warning", error.textOfError);
    }
}

//Знаходження найдорожчого та найкоротшого фільму
void MainWindow::on_mostExpensiveAndShortestButton_clicked()
{
    try
    {
        QString str =
QString::fromStdString(list.mostExpensiveAndShortestByCountry(ui-
>searchCountryLineEdit->text().toStdString()));
        ui->plainTextEdit->setPlainText(str);
    }
    catch (MovieList::Errors& error)
    {
        QMessageBox::critical(this, "Error", error.textOfError);
    }
}

```

```

}

//Очищення списку фільмів
void MainWindow::on_clearListButton_clicked()
{
    list.clear();
    list.display(ui->tableWidget);
    ui->indexSpinBox->setMaximum(list.getSize());
}

//Видалення одного елементу зі списку
void MainWindow::on_deleteButton_clicked()
{
    try
    {
        list.deleteElement(ui->indexSpinBox->value());
        list.display(ui->tableWidget);
        ui->indexSpinBox->setMaximum(list.getSize());
    }
    catch (MovieList::Errors& error)
    {
        QMessageBox::warning(this, "Warning", error.textOfError);
    }
}

```

4.7 main.cpp

```

#include "mainwindow.h"

#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();
    w.setWindowTitle("Movie List Manager");
    return a.exec();
}

```

5. Протокол роботи

Програма виконує такі завдання:

- Сортування фільмів у таблиці країною виробництва.

The screenshot displays the 'Movie List Manager' application window. It features a table with movie data and a sidebar with search and sorting controls.

	Name	Director	Year	Actors	Budget(mi\$)	County	Length(min)
1	Shutter Island	Martin Scorsese	2010	Leonardo DiCaprio Mark Ruffalo Ben Kingsley Max VonSydow	80	BRA	148
2	Tini zabutykh ...	Sergei Parajanov	1965	Ivan Mikolaychuk Tatyana Bestayeva Nikolay Grinko Nina Alisova	40	UKR	125
3	Zemlya	Sergei Parajanov	1930	Stepan Shkurat Yuliya Solntseva Nikolai Nademsky Pyotr Masokha	24	UKR	120
4	Vechir na Ivan...	Yuri Ilyenko	1968	Boris Khmelniyskiy Yefim Fridman Borislav Brondukov Larisa ...	55	UKR	60
5	Prisoners	Denis Villeneuve	2013	Hugh Jackman Jake Gyllenhaa Viola Davis Maria Bello	122	USA	153
6	Dune	Denis Villeneuve	2021	Jason Momoa Oscar Isaac Rebecca Ferguson Josh Brolin	122	USA	155
7	Suicide Squad	David Ayer	2016	Will Smith Jake Gyllenhaa Margot Robbie Joel Kinnaman	175	USA	123

Search and Filter Controls:

- Name:
- Director:
- Year:
- Actors:
- Budget:
- Lenght:
- Country:
- Buttons: Clear, Add new Movie
- Enter index:
- Buttons: Clear List, Delete one
- Buttons: Read From File, Write Into File
- Buttons: Sort for Country, Same Directors Lowest Budget, Most Populat Actor, Longest Movie Per Director, Oldest Movie and The most Expensive
- Actor:
- Button: Find all Movies for Actor
- Country:
- Button: Find all Movies for County With biggest budget and Sortest Length

Рис.5.1. Сортування фільмів за країною

- Пошук назв фільмів, в яких однакові режисери та найменші бюджети одночасно.

The screenshot shows the 'Movie List Manager' application. It features a table with movie data and a sidebar with search filters. The table lists 7 movies with columns for Name, Director, Year, Actors, Budget(mil\$), County, and Length(min). The sidebar includes input fields for Name, Director, Year (set to 1500), Actors, Budget, Length, and Country, along with buttons for 'Clear', 'Add new Movie', 'Enter index' (set to 0), 'Clear List', 'Delete one', 'Read From File', and 'Write Into File'. Below the table, there are five buttons for sorting: 'Sort for Country', 'Same Directors Lowest Budget', 'Most Populat Actor', 'Longest Movie Per Director', and 'Oldest Movie and The most Expensive'. A large text box below these buttons displays the result: 'Movies with same Directors and Lowest budgets are: Zemlya, Tini zabutykh predkiv'. On the right side of the interface, there are additional filters for 'Actor' and 'Country', each with a 'Find all Movies for' button, and a button for 'Find all Movies for County With biggest budget and Sortest Length'.

	Name	Director	Year	Actors	Budget(mil\$)	County	Length(min)
1	Shutter Island	Martin Scorsese	2010	Leonardo DiCaprio Mark Ruffalo Ben Kingsley Max VonSydow	80	BRA	148
2	Tini zabutykh ...	Sergei Parajanov	1965	Ivan Mikolaychuk Tatyana Bestayeva Nikolay Grinko Nina Alisova	40	UKR	125
3	Zemlya	Sergei Parajanov	1930	Stepan Shkurat Yuliya Solntseva Nikolai Nademsky Pyotr Masokha	24	UKR	120
4	Vechir na Ivan...	Yuri Ilyenko	1968	Boris Khmelniitskiy Yefim Fridman Borislav Brondukov Larisa ...	55	UKR	60
5	Prisoners	Denis Villeneuve	2013	Hugh Jackman Jake Gyllenhaa Viola Davis Maria Bello	122	USA	153
6	Dune	Denis Villeneuve	2021	Jason Momoa Oscar Isaac Rebecca Ferguson Josh Brolin	122	USA	155
7	Suicide Squad	David Ayer	2016	Will Smith Jake Gyllenhaa Margot Robbie Joel Kinnaman	175	USA	123

Movies with same Directors and Lowest budgets are: Zemlya, Tini zabutykh predkiv

Рис.5.2. Пошук назв фільмів, в яких однакові режисери та найменші бюджети одночасно

- Пошук за заданою країною виробництва всі фільми, в яких найбільші бюджети і найменша тривалість одночасно.

The screenshot shows the 'Movie List Manager' application. It features a table with movie data and a sidebar with search filters and buttons.

	Name	Director	Year	Actors	Budget(mil\$)	County	Length(min)
1	Shutter Island	Martin Scorsese	2010	Leonardo DiCaprio Mark Ruffalo Ben Kingsley Max VonSydow	80	BRA	148
2	Tini zabutykh ...	Sergei Parajanov	1965	Ivan Mikolaychuk Tatyana Bestayeva Nikolay Grinko Nina Alisova	40	UKR	125
3	Zemlya	Sergei Parajanov	1930	Stepan Shkurat Yuliya Solntseva Nikolai Nademsky Pyotr Masokha	24	UKR	120
4	Vechir na Ivan...	Yuri Ilyenko	1968	Boris Khmelnskiy Yefim Fridman Borislav Brondukov Larisa ...	55	UKR	60
5	Prisoners	Denis Villeneuve	2013	Hugh Jackman Jake Gyllenhaa Viola Davis Maria Bello	122	USA	153
6	Dune	Denis Villeneuve	2021	Jason Momoa Oscar Isaac Rebecca Ferguson Josh Brolin	122	USA	155
7	Suicide Squad	David Ayer	2016	Will Smith Jake Gyllenhaa Margot Robbie Joel Kinnaman	175	USA	123

Search filters and buttons on the right:

- Name:
- Director:
- Year:
- Actors:
- Budget:
- Lenght:
- Country:
- Buttons: Clear, Add new Movie
- Enter index:
- Buttons: Clear List, Delete one
- Buttons: Read From File, Write Into File
- Buttons: Sort for Country, Same Directors Lowest Budget, Most Populat Actor, Longest Movie Per Director, Oldest Movie and The most Expensive
- Actor:
- Button: Find all Movies for Actor
- Country:
- Button: Find all Movies for County With biggest budget and Sortest Length

Result text box:

UKR movie with biggest budget and shortest length is Vechir na Ivana Kupala

Рис.5.3. Пошук за заданою країною виробництва всі фільми, в яких найбільші бюджети і найменша тривалість одночасно.

- Пошук для кожного режисера фільмів з найбільшою тривалістю.

Movie List Manager

	Name	Director	Year	Actors	Budget(mil\$)	County	Length(min)
1	Shutter Island	Martin Scorsese	2010	Leonardo DiCaprio Mark Ruffalo Ben Kingsley Max VonSydow	80	BRA	148
2	Tini zabutykh ...	Sergei Parajanov	1965	Ivan Mikolaychuk Tatyana Bestayeva Nikolay Grinko Nina Alisova	40	UKR	125
3	Zemlya	Sergei Parajanov	1930	Stepan Shkurat Yuliya Solntseva Nikolai Nademsky Pyotr Masokha	24	UKR	120
4	Vechir na Ivan...	Yuri Ilyenko	1968	Boris Khmelitskiy Yefim Fridman Borislav Brondukov Larisa ...	55	UKR	60
5	Prisoners	Denis Villeneuve	2013	Hugh Jackman Jake Gyllenhaa Viola Davis Maria Bello	122	USA	153
6	Dune	Denis Villeneuve	2021	Jason Momoa Oscar Isaac Rebecca Ferguson Josh Brolin	122	USA	155
7	Suicide Squad	David Ayer	2016	Will Smith Jake Gyllenhaa Margot Robbie Joel Kinnaman	175	USA	123

Search Filters:

Name:
 Director:
 Year:
 Actors:
 Budget:
 Length:
 Country:

Enter index:

Sorting Options:

Longest movies per Director are:

Yuri Ilyenko - Vechir na Ivana Kupala
 Martin Scorsese - Shutter Island
 David Ayer - Suicide Squad
 Sergei Parajanov - Tini zabutykh predkiv
 Denis Villeneuve - Dune

Actor:

Country:

Рис.5.4. Пошук для кожного режисера фільмів з найбільшою тривалістю.

- Пошук найдорожчого та найстарішого фільму одночасно.

The screenshot shows the 'Movie List Manager' application. It features a table with 8 movies, a search panel on the right, and a sorting panel at the bottom.

	Name	Director	Year	Actors	Budget(mil\$)	County	Length(min)
1	Shutter Island	Martin Scorsese	2010	Leonardo DiCaprio Mark Ruffalo Ben Kingsley Max VonSydow	80	BRA	148
2	Gossip Girl	Stephanie Savage	1924	Blake Lively Leighton Meester Penn Badgley Ed Westwick	180	FRA	144
3	Tini zabutykh ...	Sergei Parajanov	1965	Ivan Mikolaychuk Tatyana Bestayeva Nikolay Grinko Nina Alisova	40	UKR	125
4	Zemlya	Sergei Parajanov	1930	Stepan Shkurat Yuliya Solntseva Nikolai Nademsky Pyotr Masokha	24	UKR	120
5	Vechir na Ivan...	Yuri Ilyenko	1968	Boris Khmelnitskiy Yefim Fridman Borislav Brondukov Larisa ...	55	UKR	60
6	Dune	Denis Villeneuve	2021	Jason Momoa Oscar Isaac Rebecca Ferguson Josh Brolin	122	USA	155
7	Prisoners	Denis Villeneuve	2013	Hugh Jackman Jake Gyllenhaa Viola Davis Maria Bello	122	USA	153
8	Suicide Squad	David Ayer	2016	Will Smith Jake Gyllenhaa Margot Robbie Joel Kinnaman	175	USA	123

Search Panel (Right):

- Name:
- Director:
- Year:
- Actors:
- Budget:
- Lenght:
- Country:
- Buttons: Clear, Add new Movie
- Enter index: (Buttons: Clear List, Delete one)
- Buttons: Read From File, Write Into File

Sorting Panel (Bottom):

- Sort for Country
- Same Directors Lowest Budget
- Most Populat Actor
- Longest Movie Per Director
- Oldest Movie and The most Expensive

Movie Details Panel (Bottom Left):

Name: Gossip Girl
 Director: Stephanie Savage
 Year: 1924
 Actors: Blake Lively Leighton Meester Penn Badgley Ed Westwick
 Budget: 180 mil\$
 Lenght: 144 min
 Country: FRA

Search Panel (Bottom Right):

- Actor:
- Find all Movies for Actor
- Country:
- Find all Movies for County With biggest budget and Sortest Length

Рис.5.5. Пошук найдорожчого та найстарішого фільму одночасно.

- Пошук за заданим актором всі фільми, в яких він (вона) знімались.

The screenshot shows the 'Movie List Manager' application window. It features a table of movies, a search panel on the right, and a results section at the bottom.

	Name	Director	Year	Actors	Budget(mil\$)	County	Length(min)
1	Shutter Island	Martin Scorsese	2010	Leonardo DiCaprio Mark Ruffalo Ben Kingsley Max VonSydow	80	BRA	148
2	Tini zabutykh ...	Sergei Parajanov	1965	Ivan Mikolaychuk Tatyana Bestayeva Nikolay Grinko Nina Alisova	40	UKR	125
3	Zemlya	Sergei Parajanov	1930	Stepan Shkurat Yuliya Solntseva Nikolai Nademsky Pyotr Masokha	24	UKR	120
4	Vechir na Ivan...	Yuri Ilyenko	1968	Boris Khmelitskiy Yefim Fridman Borislav Brondukov Larisa ...	55	UKR	60
5	Prisoners	Denis Villeneuve	2013	Hugh Jackman Jake Gyllenhaa Viola Davis Maria Bello	122	USA	153
6	Dune	Denis Villeneuve	2021	Jason Momoa Oscar Isaac Rebecca Ferguson Josh Brolin	122	USA	155
7	Suicide Squad	David Ayer	2016	Will Smith Jake Gyllenhaa Margot Robbie Joel Kinnaman	175	USA	123

Search Panel (Right):

- Name:
- Director:
- Year:
- Actors:
- Budget:
- Lenght:
- Country:
- Buttons: Clear, Add new Movie
- Enter index:
 - Buttons: Clear List, Delete one
 - Buttons: Read From File, Write Into File

Sort Buttons (Bottom):

- Sort for Country
- Same Directors Lowest Budget
- Most Populat Actor
- Longest Movie Per Director
- Oldest Movie and The most Expensive

Results Section (Bottom):

Jake Gyllenhaa played in Prisoners, Suicide Squad

Search Filters (Bottom Right):

- Actor:
- Button: Find all Movies for Actor
- Country:
- Button: Find all Movies for County With biggest budget and Sortest Length

Рис.5.6. Пошук за заданим актором всі фільми, в яких він (вона) знімались.

- Пошук найбільш популярного актора.

The screenshot shows the 'Movie List Manager' application. It features a table of movies, a sidebar with search filters, and a main area with sorting options and a search result.

	Name	Director	Year	Actors	Budget(mil\$)	County	Length(min)
1	Shutter Island	Martin Scorsese	2010	Leonardo DiCaprio Mark Ruffalo Ben Kingsley Max VonSydow	80	BRA	148
2	Tini zabutykh ...	Sergei Parajanov	1965	Ivan Mikolaychuk Tatyana Bestayeva Nikolay Grinko Nina Alisova	40	UKR	125
3	Zemlya	Sergei Parajanov	1930	Stepan Shkurat Yuliya Solntseva Nikolai Nademsky Pyotr Masokha	24	UKR	120
4	Vechir na Ivan...	Yuri Ilyenko	1968	Boris Khmelnskiy Yefim Fridman Borislav Brondukov Larisa ...	55	UKR	60
5	Prisoners	Denis Villeneuve	2013	Hugh Jackman Jake Gyllenhaa Viola Davis Maria Bello	122	USA	153
6	Dune	Denis Villeneuve	2021	Jason Momoa Oscar Isaac Rebecca Ferguson Josh Brolin	122	USA	155
7	Suicide Squad	David Ayer	2016	Will Smith Jake Gyllenhaa Margot Robbie Joel Kinnaman	175	USA	123

Search filters on the right:

- Name:
- Director:
- Year:
- Actors:
- Budget:
- Lenght:
- Country:

Buttons: Clear, Add new Movie, Enter index: , Clear List, Delete one, Read From File, Write Into File.

Sorting options:

- Sort for Country
- Same Directors Lowest Budget
- Most Populat Actor
- Longest Movie Per Director
- Oldest Movie and The most Expensive

Search result area:

Most popular actor is Jake Gyllenhaa

Actor:

Buttons: Find all Movies for Actor, Country: , Find all Movies for County With biggest budget and Sortest Length

Рис.5.7. Пошук за заданим актором всі фільми, в яких він (вона) знімались.

6. Інструкція користувача

1. Компоненти ПЗ

Програму розроблено на мові програмування C++ [3, 4] , за допомогою фреймворку Qt [5] і може експлуатуватися на операційних системах Windows, Linux, MacOS. Під час проектування підсистем застосовувався об'єктно-орієнтований підхід до програмування.

2. Встановлення ПЗ

Для роботи програми необхідно запустити на виконання файл MovieListManager.exe

3. Системні вимоги

Програма потребує оперативної пам'яті не менше 128 МБ та об'єм вільної пам'яті не менше 16МБ.

Для коректної роботи пакету необхідна користувацька машина з процесором не менше 256 МHz.

4. Базові функції ПЗ

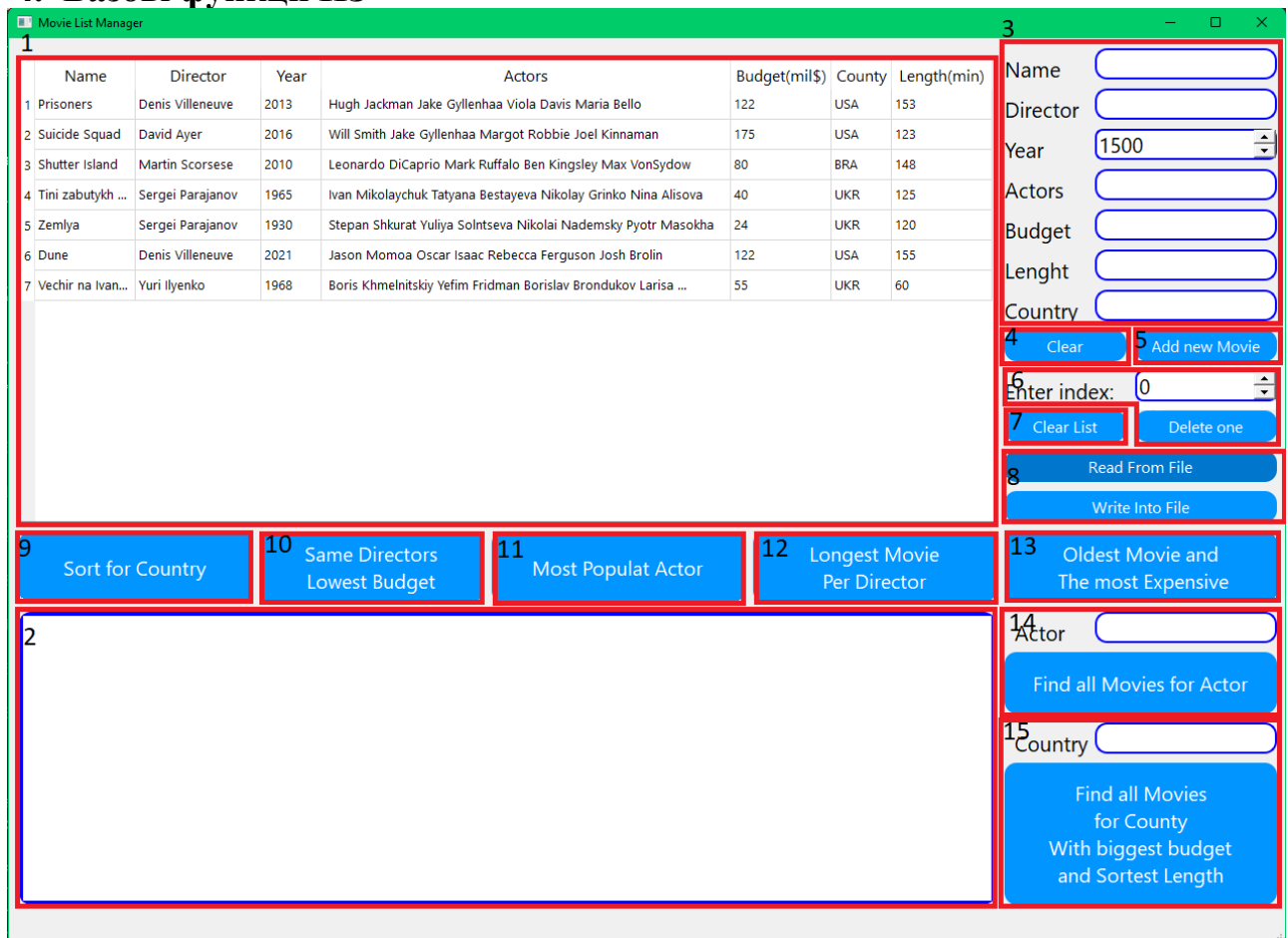


Рис.6.1. Вигляд програми

1. Таблиця фільмів
2. Текстове поле для виводу інформації
3. Поля для вводу інформації про новий фільм
4. Кнопка для очищення полів для вводу інформації(3)

5. Кнопка для додавання нового фільму з полів(3)
6. Видалення елемента за індексом
7. Кнопка для очищення всього списку фільмів
8. Кнопки для зчитування та запису в файл
9. Кнопка для сортування фільмів за країною виробництва
10. Кнопка для пошуку фільмів з однаковим режисером та найменшим бюджетом

11. Кнопка для пошуку найпопулярнішого актору
12. Кнопка для пошуку найдовшого фільму для кожного режисера
13. Кнопка для пошуку найдорожчого та найстарішого фільму
14. Пошук всіх фільмів за заданим актора
15. Пошук надорожчих та найкоротших фільмів за заданою країною виробництва

Інструкція користування програмою:

- Відкриття файлу

Для зчитування фільмів з файлу потрібно натиснути кнопку **Read From File**

- Зберігання у файл

Для запису фільмів до файлу потрібно натиснути кнопку **Write Into File**

- Додавання нового запису

Для того, щоб додати до таблиці новий фільм потрібно ввести дані про фільм та натиснути кнопку **Add new Movie**.

- Очищення полів для вводу нового фільму

Для того, щоб очистити поле для вводу нового фільму потрібно натиснути кнопку **Clear**.

- Видалення запису

Для того, щоб видалити фільм з таблиці, потрібно обрати індекс фільму та натиснути кнопку **Delete one**.

- Очищення всього списку

Для того, щоб очистити весь список фільмів потрібно натиснути кнопку **Clear List**.

- Сортування таблиці

Для того, щоб відсортувати таблицю за країною виробництва потрібно натиснути кнопку **Sort for Country**

- Пошук фільмів з однаковим режисером та найменшим бюджетом

Для пошуку фільмів з однаковим режисером та найменшим бюджетом потрібно натиснути кнопку **Same Directors Lowest Budget**

- Пошук найпопулярнішого актора

Для пошуку найпопулярнішого актора потрібно натиснути кнопку **Most Popular Actor**

- Пошук найдовшого фільму для кожного режисера

Для пошуку найдовшого фільму для кожного потрібно натиснути кнопку **Longest Movie Per Director**

- Пошук найдорожчого та найстарішого фільму

Для пошуку найдорожчого та найстарішого фільму потрібно натиснути кнопку **Oldest Movie and The most Expensive**

- Пошук всіх фільмів за заданим актора

Для пошуку всіх фільмів за заданим актора потрібно ввести ім'я актора в відповідне поле та натиснути кнопку **Find all Movie for Actor**

- Пошук найдорожчих та найкоротших фільмів за заданою країною виробництва

Для пошуку найдорожчих та найкоротших фільмів за заданою країною виробництва потрібно країну виробництва в відповідне поле та натиснути кнопку **Find all Movie for Country With biggest budget and Shortest Lenght**

- Вихід

Для виходу із програми треба натиснути на хрестик в правому верхньому углу

7. Виняткові ситуації

У програмі передбачено такі виняткові ситуації:

- Не можливо відкрити файл:

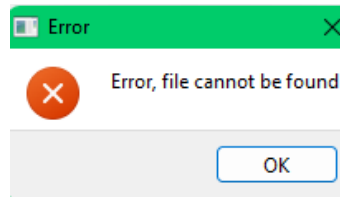


Рис.7.1. Файл не можливо відкрити

- Файл порожній:

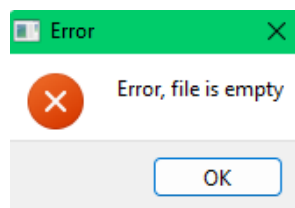


Рис.7.2. Файл порожній

- Інформація в файлі не відповідає формату:

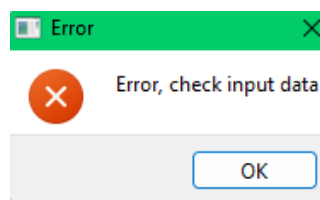


Рис.7.3. Інформація в файлі не відповідає формату

- Список фільмів порожній:

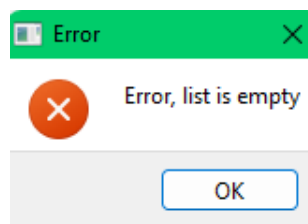


Рис.7.4. Список фільмів порожній

- Невведена інформація в полях додавання нового фільму:

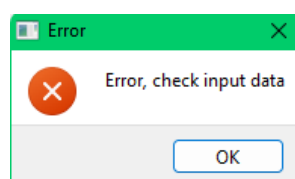


Рис.7.5. Невведена інформація в полях додавання нового фільму

- Не існує фільму який відповідає умові:

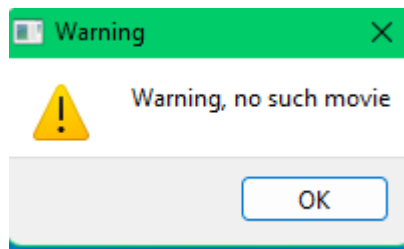
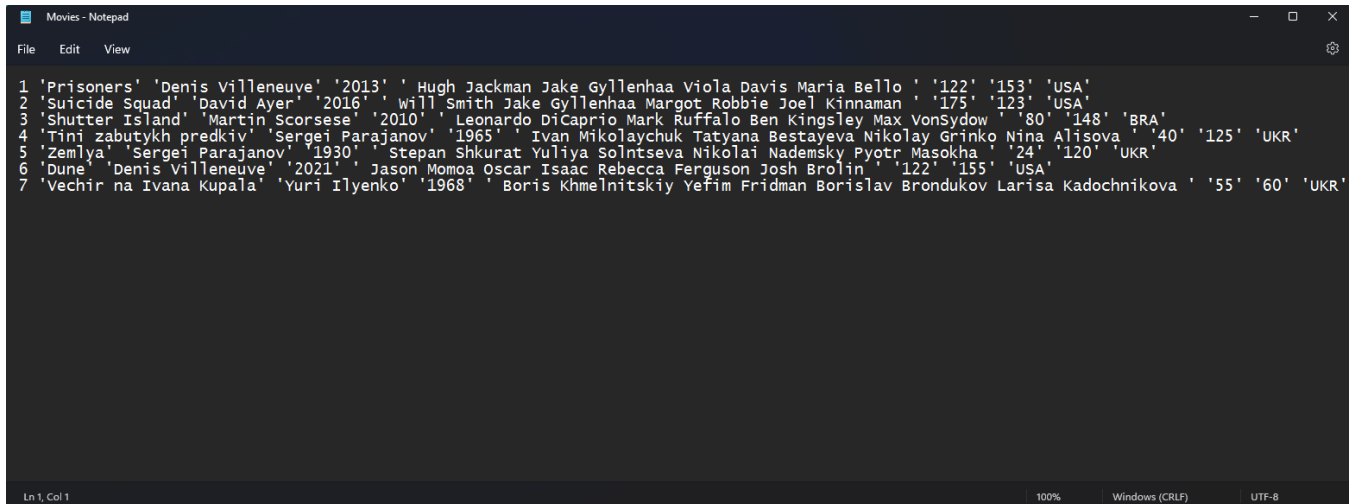


Рис.7.6. Не існує фільму який відповідає умові

8. Структура файлу вхідних даних

Файл фільмів:

Перший символ у стрічці – це порядковий номер фільму, далі через пробіл в лапках вказуються дані в такому порядку: назва, режисер, рік випуску, актори(на початку та в кінці повинний бути пробіл), бюджет, тривалість та країна виробництва фільму.



```
1 'Prisoners' 'Denis Villeneuve' '2013' ' Hugh Jackman Jake Gyllenhaal Viola Davis Maria Bello ' '122' '153' 'USA'
2 'Suicide Squad' 'David Ayer' '2016' ' Will Smith Jake Gyllenhaal Margot Robbie Joel Kinnaman ' '175' '123' 'USA'
3 'Shutter Island' 'Martin Scorsese' '2010' ' Leonardo DiCaprio Mark Ruffalo Ben Kingsley Max VonSydow ' '80' '148' 'BRA'
4 'Tini zabutykh predkiv' 'Sergei Parajanov' '1965' ' Ivan Mikolaychuk Tatyana Bestayeva Nikolay Grinko Nina Alisova ' '40' '125' 'UKR'
5 'Zemlya' 'Sergei Parajanov' '1930' ' Stepan Shkurat Yuliya Solntseva Nikolai Nademsky Pyotr Masokha ' '24' '120' 'UKR'
6 'Dune' 'Denis Villeneuve' '2021' ' Jason Momoa Oscar Isaac Rebecca Ferguson Josh Brolin ' '122' '155' 'USA'
7 'Vechir na Ivana Kupala' 'Yuri Ilyenko' '1968' ' Boris Khmelnitskiy Yefim Fridman Borislav Brondukov Larisa Kadochnikova ' '55' '60' 'UKR'
```


Висновки

При виконанні даної роботи, я розробив програму Movie List Manager, яку побудував у середовищі розробки Qt, за допомогою відповідного фреймворку. Програма створена для роботи з фільмами, які містять назву, режисера, рік видання, акторів, бюджет, країну виробництва та довжину фільму. Програма написана мовою програмування C++.

При написанні коду, я повторив стиль об'єктно-орієнтованого програмування: використовував різні класи та їхні методи. До написаних класів входять такі класи: Movie, MovieList.

Список використаної літератури

1. Об'єктно-орієнтоване програмування: методичні вказівки до виконання курсових робіт для студентів напрямку 6.121 «Інженерія програмного забезпечення» / Укл. Коротєєва Т.О., Дяконюк Л.М.— Львів: Видавництво Національного університету “Львівська політехніка”, 2020. — с.
2. Grokking Algorithms /А. Бхаргава / 2019.-288с.
3. Об'єктно-орієнтоване програмування в С++ / Р.Лафоре - Спб.: Пітер, 2004.-924 с.
4. С/С++. Програмування на мові високого рівня / Т. А. Павловская. - Спб. Пітер, 2003. —461 с.
5. Qt 5.10 Професійне програмування на С++ / М. Шлее / 2018.- 1072с.