**КОЛОМИЙСЬКИЙ ПОЛІТЕХНІЧНИЙ КОЛЕДЖ**

**НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»**

**Циклова комісія інженерія програмного забезпечення**

**КУРСОВА РОБОТА**

з дисципліни:

*"Об`єктно-орієнтоване програмування"*

на тему: "Автоматизоване робоче місце менеджера кінотеатру"

Студентки 3 курсу групи П-31

галузі знань 12

спеціальності 121

Сайко Г. Т.

(прізвище та ініціали)

Керівник: Красничук В.В.

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Оцінка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Члени комісії: \_\_\_\_\_\_\_\_ Ляшеник А.В.

(підпис)

\_\_\_\_\_\_\_\_ Смиковчук Т.В.

(підпис)

\_\_\_\_\_\_\_\_ Красничук В. В.

(підпис)

м. Коломия – 2019 рік

**КОЛОМИЙСЬКИЙ ПОЛІТЕХНІЧНИЙ КОЛЕДЖ**

**НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»**

Відділення: ***денне відділення*** ***комп’ютерних технологій***

Циклова комісія ***інженерії програмного забезпечення***

Освітньо-кваліфікаційний рівень ***молодший спеціаліст***

Галузь знань ***12 " Інформаційні технології "***

Спеціальність ***121 "Інженерія програмного забезпечення "***

**ЗАТВЕРДЖУЮ**

Голова циклової комісії інженерії програмного забезпечення

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Т.Т.Дуб**

“\_\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2019 року

**ЗАВДАННЯ**

**НА КУРСОВУ РОБОТУ**

Студентці: Сайко Галині Тарасівні

1. Тема роботи: АРМ менеджера кінотеатру

Керівник роботи: Красничук Вікторія Вікторівна

1. Термін виконання студентом роботи 18 червня 2019 року.
2. Вихідні дані до роботи: Мова програмування С++
3. Зміст розрахунково-пояснювальної записки:

1. Аналіз технічного завдання.

2. Обґрунтування алґоритму й структури програми.

3. Розроблення програми.

4. Тестування програми і результати її виконання.

Дата видачі завдання: 21.01.2019

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| №  п/п | Найменування етапів  курсової роботи | Термін виконання етапів роботи | Примітки |
| 1. | Постановка задач, аналіз специифікації вимог. | 21.01.2019 |  |
| 2. | Визначення структури даних та методів рішення. | 04.02.2019 |  |
| 3. | Визначення структури програмних модулів. | 18.02.2019 |  |
| 4. | Розробка системи класів. | 27.02.2019 |  |
| 5. | Розробка методів. | 04.03.2019 |  |
| 6. | Інтерфейс програми. | 13.03.2019 |  |
| 7. | Розроблення програми. | 20.03.2019 |  |
| 8. | Розробка документації. | 08.04.2019 |  |
| 9. | Тестування. | 10.04.2019 |  |
| 10. | Оформлення пояснювальної записки. | 24.04.2019 |  |
| 11. | Виправлення недоліків. | 13.04.2019 |  |
| 12. | Захист. | 18.06.2019 |  |

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Сайко Галина Тарасівна

(підпис студента) (прізвище, ім‘я, по батькові студента)

Керівник \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Красничук В. В.

(підпис викладача) (прізвище, ім‘я, по батькові викладача)

**Зміст**

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

3

418.КР 121.01.16 ПЗ

Розроб.

Сайко Г.Т.

Перевір.

Красничук В.В.

Реценз.

Н. Контр.

Затверд.

Пояснювальна записка.

Автоматизоване робоче місце менеджера кінотеатру

Літ.

Аркушів

*41*

КПК НУ «ЛП»

Вступ………………………………………………………………………..…….5

1 Аналіз специфікації вимог технічного завдання………….…………………6

2 Обґрунтування алґоритму й структури програми ………………………….8

3 Процес розроблення програми……..…………………….……..…………..11

3.1 Розробка системи класів…………………………………………………11

3.2 Розробка методів…………………………………..…………………….16

3.3 Опис файлів даних та інтерфейсу програми……………………………18

4 Тестування програми і результати її виконання……………………………22

Висновки..……………..……….…………………………………………..…….23

Перелік використаних джерел...…………………………………………..……24

Додаток А ...…….………………...………………………………..…………….25

Додаток Б...…….…………………….……………………………..…………….39

**Вступ**

Величезна кількість людей будь-якого віку відвідує кінотеатри. Через те що існує велика кількість фільмів, сеансів, різна ціна квитків і різний формат фільму потрібно розробити програму, яка значно полегшить роботу менеджера кінотеатру.

Метою даної курсової роботи буде створити програму, яка дає змогу записувати, змінювати, переглядати, видаляти дані щодо фільмів, які фільми будуть іти в кінотеатрі, тривалість і формат фільму, яка ціна на квитки та можливість забронювати квиток.

**1 Аналіз специфікації вимог технічного завдання**

Проаналізувавши дане завдання, я визначила мету створення програмного забезпечення, класи та функції, які потрібно використати.

Під час обґрунтування розв’язку цієї задачі, я зрозуміла, що саме головне в програмі для користувача це вигляд програми у консольному меню. Тобто, коли користувач запускає дану програму, він повинен бачити чітко реалізоване головне меню. Коли меню є візуально зрозумілим, працювати з програмою стає набагато легше.

Розробити програму, за допомогою якої можна реалізувати дані завдання:

1. Додати в базу фільм з такими параметрами:

1.1. Назва фільму.

1.2. Тривалість.

1.3. Формат (2D, 3D).

1.4. Жанр.

1.5. Дата виходу фільму.

2. Пошук фільму за назвою, жанрем, тривалістю, форматом та датою.

3. Додати в базу сеанс з такими параметрами:

3.1. Номер сеансу.

3.2. Назва фільму.

3.3. Формат залу.

3.4. Дата і час.

3.5. Ціна квитку.

4. Переглянути всі фільми, сеанси та місця в залі.

5. Забронювати квиток.

6. Пошук сеансу за датою, часом та форматом.

7. Видалити фільм або сеанс.

8. Додати в базу працівників за такими параметрами:

8.1. Ідентифікаційний код.

8.2. П.І.Б.

8.3. Посада.

8.4. Зарплата.

8.5. Місце проживання.

8.6. Дата народження.

8.7. Зміна.

9. Пошук працівників за іменем, посадою, кодом та зміною.

10. Показати всіх працівників.

11. Видалити працівника.

12. Вихід з програми.

В моєму випадку дані які я буду заносити або виводити на консоль будуть зберігатись у текстових файлах.

**2 Обґрунтування алґоритму й структури програми**

Базові принципи ООП:

**Клас**

Клас визначає абстрактні характеристики деякої сутності, включаючи характеристики самої сутності (її атрибути або властивості) та дії, які вона здатна виконувати (її поведінки, методи або можливості). Наприклад, клас може характеризуватись даними, потрібними для купівлі квитка, зокрема: код сеансу, ряд, місце. Класи вносять модульність та структурованість в об'єктно-орієнтовану програму. Як правило, клас має бути зрозумілим для не-програмістів, що знаються на предметній області, що, у свою чергу, значить, що клас повинен мати значення в контексті. Також, код реалізації класу має бути досить самодостатнім. Властивості та методи класу, разом називаються його членами.

**Об'єкт**

Окремий екземпляр класу (створюється після запуску програми і ініціалізації полів класу). Клас відповідає всій інформації про сеанси та їх фільми. Об'єкт є одним окремим фільмом, окремим варіантом значень характеристик. Має назву, має тривалість, жанр та формат. Об'єкт є екземпляром (примірником) класу. Сукупність значень атрибутів окремого об'єкта називається станом. На основі класу можна, також, створити інший об'єкт, який відрізнятиметься від об'єкта своїм станом (наприклад назвою). Обидва об'єкта є екземплярами класу.

**Метод**

Можливості об'єкта. Тому отримати\_імя() є одним із методів об'єкта. Він може мати й інші методи, зокрема: отримати\_жанр(), або отрима-ти\_формат(). В межах програми, використання методу має впливати лише на один об'єкт, всі можуть мати ім’я, але треба отримати одне окреме замовлення.

ООП характеризується трьома базовими принципами:

1)Інкапсуляція

2)Наслідування

3)Поліморфізм

**Інкапсуляція**

Інкапсуляція являє собою комбінування даних (записи, структури) з процедурами та функціями для отримання нового типу даних.

Тут проводиться аналогія з фізичними об'єктами. Конкретні фізичні властивості визначаються даними різних типів. Крім того, будь-який фізичний об'єкт характеризується своєю поведінкою у зовнішньому світі. Поводження об'єкта задається процедурами і функціями.

Отже, інкапсуляція означає, що методи (коди) і дані одночасно в одній і тій же структурі.

**Наслідування**

Наслідування служить для використання одного певного класу в побудові цілої ієрархії похідних класів, кожен з яких успадковує доступ до даних і методів роботи з ними всіх своїх «батьків».

Тобто, можна побудувати ієрархію класів, яка виражає родовідне дерево класів. Класи організовані в єдину деревоподібну структуру з загальним коренем. Властивості і методи певного класу автоматично доступні будь-якого класу, розташованому нижче в ієрархічному дереві.

Таким чином, спадкування – це особливість ООП, за допомогою якої клас може бути визначений як розширення вже існуючого класу.

Тут клас «Сеанс» успадковує всі властивості і методи попередньо описаних класів «Зал» та «Фільм», тобто для нього визначені їх поля та методи. Крім того, клас «Сеанс» володіє власними методами.

Успадкування дозволяє різним типам даних спільно використовувати один і той же код, приводячи до зменшення його розміру та підвищення функціональності.

**Поліморфізм**

Поліморфізм означає залежність поведінки від класу, в якому ця поведінка викликається, тобто, два або більше класів можуть реагувати по-різному на однакові повідомлення. Наприклад, якщо отримує команду ціна(), то у відповідь можна отримати ціну замовлення; якщо отримує команду ціна (), то у відповідь можна отримати зарплату. На практиці - це реалізовується шляхом реалізації ряду підпрограм (функцій, процедур, методи тощо) з однаковими іменами, але з різними параметрами. В залежності від того, що передається і вибирається відповідна підпрограма.

**3 ПРОЦЕС РОЗРОБЛЕННЯ ПРОГРАМИ**

**3.1 Розробка системи класів**

Розробка даної програми здійснювалася на мові програмування С++, за допомогою програми DevC++.

В програмі є 4 основних класи. Фільм, Працівник, Зал, Сеанс. Також є 3 Каталоги, каталог фільмів, каталог сеансів, каталог працівників, крім Залу, які зберігають перелік об'єктів відповідного класу в контейнер для зберігання даних. В якості контейнера даних для зберігання об'єктів використовується вектор.

Клас Зал реалізований для роботи з залом кінотеатру. В конструкторі ініціалізується двохвимірний масив типу bool, який відповідає за вільні та зайняті місця. Всі місця по замовчуванню вільні. Даний клас містить методи, за допомогою яких можна переглядати усі місця, окремо виділяються зайняті, також метод який дозволяє продати місце, та метод який підраховує кількість проданих місць.

В класах Фільм, Працівники, Сеанс містяться тільки гетери та сетери. А в класі їх каталогів вже міститься пошук, редагування, видалення, добавлення даних. Клас Сеанс наслідує класи Зал та Фільм.

Діаграми класів:

|  |
| --- |
| **Pracivnuk** |
| int code;  string name;  string surname;  string posada;  string data\_narodzhennya;  string vulutsya\_prozhuvannya;  float zarplata;  int zmina; |
| Pracivnuk();  +Pracivnuk(int code);  +void set\_data();  +void set\_code(int code);  +void set\_name(string name);  +void set\_surname(string surname);  +void set\_posada(string posada);  +void set\_data\_narodzhennya(string data\_narodzhennya);  +void set\_vulutsya\_prozhuvannya(string vulutsya\_prozhuvannya);  +void set\_zarplata(float zarplata);  +void set\_zmina(int zmina);  +int get\_code();  +string get\_name();  +string get\_surname();  +string get\_posada();  +string get\_data\_narodzhennya();  +string get\_vulutsya\_prozhuvannya();  +float get\_zarplata();  +int get\_zmina(); |

|  |
| --- |
| **Zal** |
| unsigned int kilkist\_ryadiv;  unsigned int miscya\_in\_ryad;  bool \*\*miscya; |
| Zal();  +void show\_miscya();  +void buy\_misce(int i, int j);  +int kilkist\_prodanuh\_kvutkiv();  +void set\_misce(int i, int j, bool isBuy);  +bool get\_misce(int i, int j);  +int get\_kilkist\_ryadiv();  +int get\_miscya\_in\_ryad(); |

|  |
| --- |
| **Film** |
| int code;  string nazva;  string zhanr;  string data\_vuhoda;  int truvalist;  char format[3]; |
| Film();  Film(int code);  +void set\_data();  +void set\_code(int code);  +void set\_nazva(string nazva);  +void set\_zhanr(string zhanr);  +void set\_data\_vuhoda(string data\_vuhoda);  +void set\_truvalist(int truvalist);  +void set\_format(char format[3]);  +int get\_code();  +string get\_nazva();  +string get\_zhanr();  +string get\_data\_vuhoda();  +int get\_truvalist();  +char\* get\_format(); |

|  |
| --- |
| **Seans : public Zal, public Film** |
| int code;  string date;  string chas;  int cina\_kvutka;  int zmina; |
| Seans(Film film);  Seans(int code);  Seans(Film film, int id);  +void set\_data();  +void set\_code(int code);  +void set\_date(string date);  +void set\_chas(string chas);  +void set\_cina\_kvutka(int cina\_kvutka);  +void set\_zmina(int zmina);  +int get\_code();  +string get\_date();  +string get\_chas();  +int get\_cina\_kvutka();  +int get\_zmina();  +void edit\_film(Film film); |

|  |
| --- |
| **class CataloguePracivnukiv** |
| vector<Pracivnuk> catalogue;  vector<Pracivnuk>::iterator iterCatalogue; |
| CataloguePracivnukiv();  +void in\_file();  +void out\_file();  +void menu();  +bool check\_id(const Pracivnuk &pracivnuk);  +void add\_pracivnuk();  +void search\_pracivnuk();  +void delete\_pracivnuk();  +void edit\_pracivnuk();  +void show\_all();  +void search\_by\_zmina(int szmina); |

|  |
| --- |
| **class CatalogueSeans** |
| vector<Seans> catalogue;  vector<Seans>::iterator iterCatalogue; |
| CatalogueSeans();  +void in\_file();  +void out\_file();  +void menu();  +bool check\_id(const Seans &seans);  +void add\_seans();  +void search\_seans();  +void delete\_seans();  +void edit\_seans();  +void show\_all();  +void show\_miscya();  +void buy\_misce();  +void show\_zmina();  +void profit(); |

|  |
| --- |
| **class CatalogueFilms** |
| vector<Film> catalogue;  vector<Film>::iterator iterCatalogue; |
| CatalogueFilms();  +void in\_file();  +void out\_file();  +void menu();  +bool check\_id(const Film &film);  +void add\_film();  +void search\_film();  +void delete\_film();  +void edit\_film();  +void show\_all();  +Film get\_by\_id(int id); |

**3.2 Розробка методів**

В програмі існують три класи каталоги об'єктів CataloguePracivnuk, CatalogueFilm, CatalogueSeans. Кожен з них містить контейнер даних класу вектор, де зберігаються відповідні об'єкти: в каталозі працівників - об'єкти класу Pracivnuk, в каталозі фільмів - об'єкти класу Film, в каталозі сеансів - каталог об'єкту Seans.

У зв'язку з тим, що ці класи майже однакові, для прикладу візьмемо CataloguePracivnuk.

Для початку необхідно було створити метод, який би відповідав за добавлення нового об'єкта в контейнер. Метод називається add\_pracivnuk(). В цьому методі створюється об'єкт класу Pracivnuk, за допомогою конструктора заповнюються поля. Потім перевіряється id на унікальність, якщо це так, то новий об'єкт додається в кінець вектору. Реалізація даного методу в Додатку А.

Дальше були необхідні методи запису в файл та зчитування з нього. Їх реалізацію показано в Додатку А.

Потім йшли методи редагування даних, їх пошук та видалення. Пошук даних здійснювався за допомогою ітератора, послідовно перебираючи всі елементи вектору. Приклади цих методів також можна переглянути в Додатку А.

Щодо класу Zal, в ньому реалізовані такі основні методи:

- вивід всіх місць - void show\_miscya(). Зайняті місця виводяться в квадратних дужках

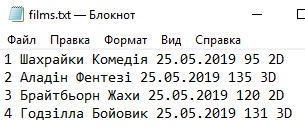
- купити місце - void buy\_misce(int i, int j);. Параметри в даному методі, це номер ряду та номер місця.

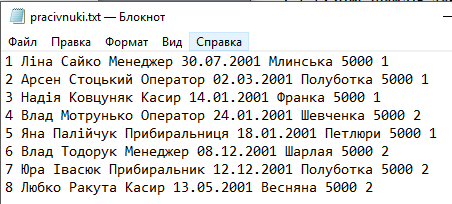
- підрахунок проданих місць - int kilkist\_prodanuh\_kvutkiv(); (необхідно для підрахунку виручки)

Приклад даних методів в додатку A.

**3.3 Опис файлів даних та інтерфейсу програми**

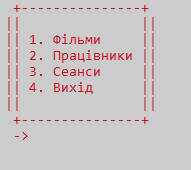
Програма використовує файлові об’єкти для зберігання проміжних чи кінцевих результатів роботи у текстовому форматі.



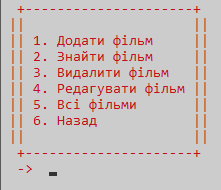


**Консоль:**

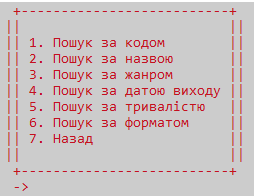
Головне меню:



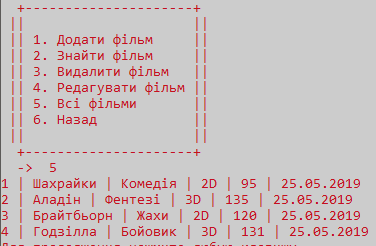
Пункт меню Фільми:



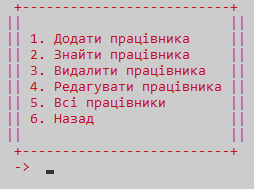
В ньому ми можемо додати новий фільм, видалити фільм, редагувати, вивести на екран всі фільми та здійснити пошук фільму за такими параметрами:



Пункт меню Всі фільми:

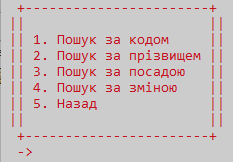


Пункт меню Працівники, на головному меню:

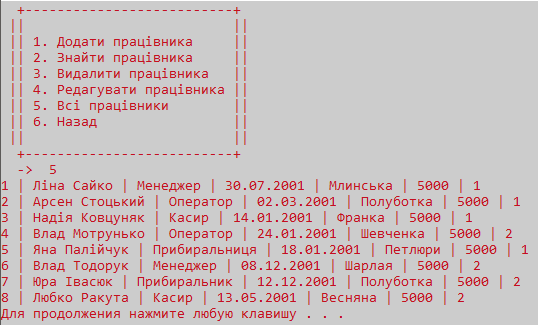


Аналогічно і тут, можемо видалити, додати, редагувати, знайти, вивести список всіх працівників.

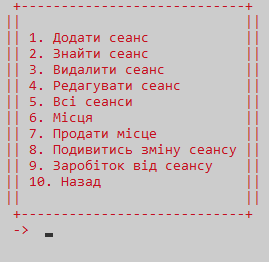
Пошук здійснюється за такими параметрами:



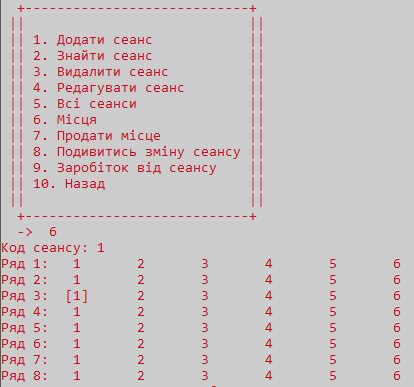
Пункт меню Всі працівники:



Пункт меню Сеанси, на головному меню:

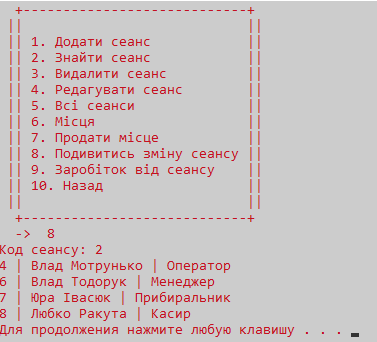


Пункт меню Місця:



Тут ми можемо бачити всі місця які є в залі. Зайняті місця позначаються в квадратних дужках [1].

Пункт меню Подивитись зміну сеансу:



В кінотеатрі є 2 зміни. Ті хто працюють в 1 зміні, виходять на роботу по непарних днях. Ті хто в 2, по парних.

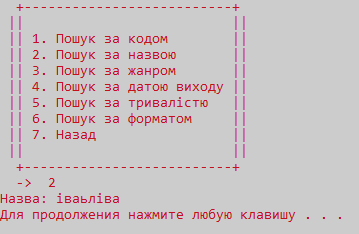
**4 Тестування програми і результати її виконання**

Суть тестування програми полягає в тому, щоб перевірити її на наявні помилки. Якщо такі існують, необхідно їх виправити.

Після кожного етапу розробки проводилося невелике тесутвання, результатом якого було знаходження проблем та подальше їх усунення.

В результаті фінального тестування помилок знайдено не було, все працює правильно, всі кнопки виконують поставлені задачі, всі дані відображаються коректно.

При помилці, яку допускає користувач, програма пропонує натиснути для продовження будь-яку клавішу. Наприклад, коли ми при пошуку фільму за назвою, вводимо назву неіснуючого фільму:



**Висновки**

В даній курсовій роботі була організована програма, яка зберігає дані у текстовий файл, зчитує з неї та виводить дані на екран. Програма була написані у середовищі DevC++.

Виконуючи курсову роботу я навчилася правильно та ефективно використовувати принципи ООП. Вдосконалила свої навички у роботі з файлами. Використовувала стандартні шаблони бібліотеки stl. Та в загальному вдосконалила свої навички у об’єктно орієнтованому програмуванні.

**Література:**

1. Бьярне Страуструп "Программирование принципы и практика использования C++".

2. Герберт Шилдт: С++ базовый курс.

**Інформаційні джерела:**

1. www.wikipedia.org

2. [www.cppstudio.com](http://www.cppstudio.com)

3. <https://code-live.ru>

**Додаток А**

Нижче знаходяться коди деяких основних методів програми.

**Код виконання:**

* cataloguefilms.cpp

CatalogueFilms::CatalogueFilms() {

out\_file();

}

*//метод для запису даних у файл. Викликається кожен раз коли відбуваються якісь зміни (видалення, редагування чи додавання інформації), за подібним принципом заносимо й інші дані для інших класів в інші файли*

void CatalogueFilms::in\_file() {

ofstream film\_file("films.txt", ios\_base::trunc);

for(iterCatalogue = catalogue.begin(); iterCatalogue != catalogue.end(); \*iterCatalogue++) {

film\_file << iterCatalogue->get\_code() << " " <<

iterCatalogue->get\_nazva() << " " <<

iterCatalogue->get\_zhanr() << " " <<

iterCatalogue->get\_data\_vuhoda() << " " <<

iterCatalogue->get\_truvalist() << " " <<

iterCatalogue->get\_format() << " " << endl;

}

film\_file.close();

}

*//метод для виводу даних з файлу. Викликається один раз, при створенні каталогу*

void CatalogueFilms::out\_file() {

int code;

string nazva;

string zhanr;

string data\_vuhoda;

int truvalist;

char format[3];

Film film(-1);

ifstream film\_file("films.txt");

while(film\_file >> code >> nazva >> zhanr >> data\_vuhoda >> truvalist >> format) {

film.set\_code(code);

film.set\_nazva(nazva);

film.set\_zhanr(zhanr);

film.set\_data\_vuhoda(data\_vuhoda);

film.set\_truvalist(truvalist);

film.set\_format(format);

catalogue.push\_back(film);

}

film\_file.close();

}

*//метод для перевірки існує введене id чи ні*

bool CatalogueFilms::check\_id(const Film &film) {

for(iterCatalogue = catalogue.begin(); iterCatalogue != catalogue.end(); \*iterCatalogue++) {

if(iterCatalogue->code == film.code) {

return false;

}

}

return true;

}

*//метод для додавання нового фільму*

void CatalogueFilms::add\_film() {

Film film;

if(check\_id(film)) {

catalogue.push\_back(film);

}

else {

cout << "Такий код вже використовується. Змініть код на інший" << endl;

}

system("pause");

}

*//метод для пошуку фільму*

void CatalogueFilms::search\_film() {

bool exit = false;

int choise;

int code;

string nazva;

string zhanr;

string data\_vuhoda;

int truvalist\_from;

int truvalist\_to;

char format[3];

while(!exit) {

system("cls");

cout << " +--------------------------+" <<

"\n || ||" <<

"\n || " << "1. " << "Пошук за кодом " << "||" <<

"\n || " << "2. " << "Пошук за назвою " << "||" <<

"\n || " << "3. " << "Пошук за жанром " << "||" <<

"\n || " << "4. " << "Пошук за датою виходу " << "||" <<

"\n || " << "5. " << "Пошук за тривалістю " << "||" <<

"\n || " << "6. " << "Пошук за форматом " << "||" <<

"\n || " << "7. " << "Назад " << "||" <<

"\n || ||" <<

"\n +--------------------------+" <<

"\n " << "-> ";

cin >> choise;

switch(choise) {

case 1:

cout << "Код: ";

cin >> code;

cout << endl;

for(iterCatalogue = catalogue.begin(); iterCatalogue != catalogue.end(); \*iterCatalogue++) {

if(iterCatalogue->get\_code() == code) {

cout << \*iterCatalogue;

}

}

break;

case 2:

cout << "Назва: ";

cin >> nazva;

for(iterCatalogue = catalogue.begin(); iterCatalogue != catalogue.end(); \*iterCatalogue++) {

if(iterCatalogue->get\_nazva() == nazva) {

cout << iterCatalogue->get\_code() << " | " << iterCatalogue->get\_nazva() << " | " << iterCatalogue->get\_zhanr() << " | " << iterCatalogue->get\_format() << " | " << iterCatalogue->get\_truvalist() << " | " << iterCatalogue->get\_data\_vuhoda() << endl;

}

}

break;

case 3:

cout << "Жанр: ";

cin >> zhanr;

for(iterCatalogue = catalogue.begin(); iterCatalogue != catalogue.end(); \*iterCatalogue++) {

if(iterCatalogue->get\_zhanr() == zhanr) {

cout << iterCatalogue->get\_code() << " | " << iterCatalogue->get\_nazva() << " | " << iterCatalogue->get\_zhanr() << " | " << iterCatalogue->get\_format() << " | " << iterCatalogue->get\_truvalist() << " | " << iterCatalogue->get\_data\_vuhoda() << endl;

}

}

break;

case 4:

cout << "Дата виходу (dd.mm.yyyy): ";

cin >> data\_vuhoda;

for(iterCatalogue = catalogue.begin(); iterCatalogue != catalogue.end(); \*iterCatalogue++) {

if(iterCatalogue->get\_data\_vuhoda() == data\_vuhoda) {

cout << iterCatalogue->get\_code() << " | " << iterCatalogue->get\_nazva() << " | " << iterCatalogue->get\_zhanr() << " | " << iterCatalogue->get\_format() << " | " << iterCatalogue->get\_truvalist() << " | " << iterCatalogue->get\_data\_vuhoda() << endl;

}

}

break;

case 5:

cout << "Тривалість (hv)\n від: ";

cin >> truvalist\_from;

cout << " до: ";

cin >> truvalist\_to;

for(iterCatalogue = catalogue.begin(); iterCatalogue != catalogue.end(); \*iterCatalogue++) {

if(iterCatalogue->get\_truvalist() >= truvalist\_from && iterCatalogue->get\_truvalist() <= truvalist\_to) {

cout << iterCatalogue->get\_code() << " | " << iterCatalogue->get\_nazva() << " | " << iterCatalogue->get\_zhanr() << " | " << iterCatalogue->get\_format() << " | " << iterCatalogue->get\_truvalist() << " | " << iterCatalogue->get\_data\_vuhoda() << endl;

}

}

break;

case 6:

cout << "Формат: ";

cin >> format;

for(iterCatalogue = catalogue.begin(); iterCatalogue != catalogue.end(); \*iterCatalogue++) {

if(!strcmp(iterCatalogue->get\_format(), format)) {

cout << iterCatalogue->get\_code() << " | " << iterCatalogue->get\_nazva() << " | " << iterCatalogue->get\_zhanr() << " | " << iterCatalogue->get\_format() << " | " << iterCatalogue->get\_truvalist() << " | " << iterCatalogue->get\_data\_vuhoda() << endl;

}

}

break;

case 7:

exit = true;

}

system("pause");

}

}

void CatalogueFilms::delete\_film() {

int code;

cout << "Код: ";

cin >> code;

for(unsigned int i = 0; i < catalogue.size(); i++) {

if(catalogue[i].get\_code() == code) {

catalogue.erase(catalogue.begin() + i);

return;

}

}

cout << "Код не знайдено" << endl;

system("pause");

}

*//метод для редагування даних про фільм*

void CatalogueFilms::edit\_film() {

bool exit = false;

int choise;

bool isPresent = false;

int code;

string nazva;

string zhanr;

string data\_vuhoda;

int truvalist;

char format[3];

cout << "Код фільму який будемо редагувати: ";

cin >> code;

unsigned int i;

for(i = 0; i < catalogue.size(); i++) {

if(catalogue[i].get\_code() == code) {

isPresent = true;

break;

}

}

if(!isPresent) {

cout << "Код не знайдено" << endl;

return;

}

else {

while(!exit) {

system("cls");

cout << " +---------------------------+" <<

"\n || ||" <<

"\n || " << "1. " << "Редагувати назву " << "||" <<

"\n || " << "2. " << "Редагувати жанр " << "||" <<

"\n || " << "3. " << "Редагувати дату виходу " << "||" <<

"\n || " << "4. " << "Редагувати тривалість " << "||" <<

"\n || " << "5. " << "Редагувати формат " << "||" <<

"\n || " << "6. " << "Назад " << "||" <<

"\n || ||" <<

"\n +---------------------------+" <<

"\n " << "-> ";

cin >> choise;

switch(choise) {

case 1:

cout << "Нова назва: ";

cin >> nazva;

catalogue[i].set\_nazva(nazva);

break;

case 2:

cout << "Новий жанр: ";

cin >> zhanr;

catalogue[i].set\_zhanr(zhanr);

break;

case 3:

cout << "Нова дата виходу (dd.mm.yyyy): ";

cin >> data\_vuhoda;

catalogue[i].set\_data\_vuhoda(data\_vuhoda);

break;

case 4:

cout << "Нова тривалість: ";

cin >> truvalist;

catalogue[i].set\_truvalist(truvalist);

break;

case 5:

cout << "Новий формат: ";

cin >> format;

catalogue[i].set\_format(format);

break;

case 6:

exit = true;

}

system("pause");

}

}

}

* cataloguepracivnukiv.cpp

*//пошук працівника*

void CataloguePracivnukiv::search\_pracivnuk() {

bool exit = false;

int choise;

int code;

string surname;

string posada;

int zmina;

while(!exit) {

system("cls");

cout << " +-----------------------+" <<

"\n || ||" <<

"\n || " << "1. " << "Пошук за кодом " << "||" <<

"\n || " << "2. " << "Пошук за прізвищем " << "||" <<

"\n || " << "3. " << "Пошук за посадою " << "||" <<

"\n || " << "4. " << "Пошук за зміною " << "||" <<

"\n || " << "5. " << "Назад " << "||" <<

"\n || ||" <<

"\n +-----------------------+" <<

"\n " << "-> ";

cin >> choise;

switch(choise) {

case 1:

cout << "Код: ";

cin >> code;

cout << endl;

for(iterCatalogue = catalogue.begin(); iterCatalogue != catalogue.end(); \*iterCatalogue++) {

if(iterCatalogue->get\_code() == code) {

cout << \*iterCatalogue;

}

}

break;

case 2:

cout << "Пріщвище: ";

cin >> surname;

for(iterCatalogue = catalogue.begin(); iterCatalogue != catalogue.end(); \*iterCatalogue++) {

if(iterCatalogue->get\_surname() == surname) {

cout << iterCatalogue->get\_code() << " | " << iterCatalogue->get\_name() << " " << iterCatalogue->get\_surname() << " | " << iterCatalogue->get\_posada() << " | " << iterCatalogue->get\_data\_narodzhennya() << " | " << iterCatalogue->get\_vulutsya\_prozhuvannya() << " | " << iterCatalogue->get\_zarplata() << " | " << iterCatalogue->get\_zmina() << endl;

}

}

break;

case 3:

cout << "Посада: ";

cin >> posada;

for(iterCatalogue = catalogue.begin(); iterCatalogue != catalogue.end(); \*iterCatalogue++) {

if(iterCatalogue->get\_posada() == posada) {

cout << iterCatalogue->get\_code() << " | " << iterCatalogue->get\_name() << " " << iterCatalogue->get\_surname() << " | " << iterCatalogue->get\_posada() << " | " << iterCatalogue->get\_data\_narodzhennya() << " | " << iterCatalogue->get\_vulutsya\_prozhuvannya() << " | " << iterCatalogue->get\_zarplata() << " | " << iterCatalogue->get\_zmina() << endl;

}

}

break;

case 4:

cout << "Зміна: ";

cin >> zmina;

for(iterCatalogue = catalogue.begin(); iterCatalogue != catalogue.end(); \*iterCatalogue++) {

if(iterCatalogue->get\_zmina() == zmina) {

cout << iterCatalogue->get\_code() << " | " << iterCatalogue->get\_name() << " " << iterCatalogue->get\_surname() << " | " << iterCatalogue->get\_posada() << " | " << iterCatalogue->get\_data\_narodzhennya() << " | " << iterCatalogue->get\_vulutsya\_prozhuvannya() << " | " << iterCatalogue->get\_zarplata() << " | " << iterCatalogue->get\_zmina() << endl;

}

}

break;

case 5:

exit = true;

}

system("pause");

}

}

*//видалення працівника*

void CataloguePracivnukiv::delete\_pracivnuk() {

int code;

cout << "Код: ";

cin >> code;

for(unsigned int i = 0; i < catalogue.size(); i++) {

if(catalogue[i].get\_code() == code) {

catalogue.erase(catalogue.begin() + i);

return;

}

}

cout << "Код не знайдено" << endl;

system("pause");

}

*//пошук працівника за зміною*

void CataloguePracivnukiv::search\_by\_zmina(int szmina) {

for(iterCatalogue = catalogue.begin(); iterCatalogue != catalogue.end(); \*iterCatalogue++) {

if(iterCatalogue->get\_zmina() == szmina) {

cout << iterCatalogue->get\_code() << " | " << iterCatalogue->get\_name() << " " << iterCatalogue->get\_surname() << " | " << iterCatalogue->get\_posada() << endl;

}

}

system("pause");

}

* catalogueseans.срр

*//метод для виводу на екран місць*

void CatalogueSeans::show\_miscya() {

int code;

cout << "Код сеансу: ";

cin >> code;

for(unsigned int i = 0; i < catalogue.size(); i++) {

if(catalogue[i].get\_code() == code) {

catalogue[i].show\_miscya();

return;

}

}

cout << "Код не знайдено" << endl;

system("pause");

}

*//метод для покупки місця*

void CatalogueSeans::buy\_misce() {

int code;

int ryad, misce;

cout << "Код сеансу: ";

cin >> code;

for(unsigned int i = 0; i < catalogue.size(); i++) {

if(catalogue[i].get\_code() == code) {

cout << "Ряд: ";

cin >> ryad;

cout << "Місце: ";

cin >> misce;

if(!catalogue[i].get\_misce(ryad-1, misce-1)) {

cout << "Місце зайнято" << endl;

system("pause");

return;

}

catalogue[i].buy\_misce(ryad, misce);

return;

}

}

cout << "Код не знайдено" << endl;

system("pause");

}

*//метод для підрахунку прибутку*

void CatalogueSeans::profit() {

int suma = 0;

int code;

cout << "Код сеансу: ";

cin >> code;

for(unsigned int i = 0; i < catalogue.size(); i++) {

if(catalogue[i].get\_code() == code) {

suma = catalogue[i].get\_cina\_kvutka() \* catalogue[i].kilkist\_prodanuh\_kvutkiv();

cout << "Прибуток = " << suma << endl;

system("pause");

return;

}

}

cout << "Код не знайдено" << endl;

system("pause");

}

* film.cpp

//метод для виводу інформації про фільм на екран

ostream& operator<<(std::ostream &out, Film film) {

out << "Код: " << film.code <<

"\nНазва: " << film.nazva <<

"\nЖанр: " << film.zhanr <<

"\nДата виходу: " << film.data\_vuhoda <<

"\nТривалість (хв): " << film.truvalist <<

"\nФормат: " << film.format << endl;

return out;

}

* main.cpp

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

string setColor(int i) {

SetConsoleTextAttribute(hConsole, (WORD)((7 << 4) | i));

return "";

}

int main() {

system("chcp 1251 > nul");

CatalogueFilms catalogueFilms;

CataloguePracivnukiv cataloguePracivnukiv;

CatalogueSeans catalogueSeans;

bool exit = false;

int choise;

system("color 70");

while(!exit) {

system("cls");

cout << setColor(4) << " +---------------+" <<

"\n || ||" <<

"\n || " << "1. " << "Фільми " << "||" <<

"\n || " << "2. " << "Працівники " << "||" <<

"\n || " << "3. " << "Сеанси " << "||" <<

"\n || " << "4. " << "Вихід " << "||" <<

"\n || ||" <<

"\n +---------------+" <<

"\n " << "-> ";

COORD pos = {5, 8};

SetConsoleCursorPosition ( hConsole, pos );

cin >> choise;

switch(choise) {

case 1:

catalogueFilms.menu();

break;

case 2:

cataloguePracivnukiv.menu();

break;

case 3:

catalogueSeans.menu();

break;

case 4:

exit = true;

}

}

return 0;

}

* zal.cpp

*//ініціалізація двохвимірного динамічного масиву*

Zal::Zal() {

kilkist\_ryadiv = 8;

miscya\_in\_ryad = 6;

miscya = new bool\*[kilkist\_ryadiv];

for(unsigned int i = 0; i < kilkist\_ryadiv; i++)

miscya[i] = new bool[miscya\_in\_ryad];

for(unsigned int i = 0; i < kilkist\_ryadiv; i++) {

for(unsigned int j = 0; j < miscya\_in\_ryad; j++) {

miscya[i][j] = true;

}

}

}

*//метод для виводу місць*

void Zal::show\_miscya() {

int misce;

for(unsigned int i = 0; i < kilkist\_ryadiv; i++) {

cout << "Ряд " << i+1 << ":\t";

misce = 0;

for(unsigned int j = 0; j < miscya\_in\_ryad; j++) {

misce++;

if(!miscya[i][j]) {

cout << "[" << misce << "]" << "\t";

}

else cout << " " << misce << "\t";

}

cout << endl;

}

system("pause");

}

*//метод для підрахунку кількості проданих квитків*

int Zal::kilkist\_prodanuh\_kvutkiv() {

int count = 0;

for(unsigned int i = 0; i < kilkist\_ryadiv; i++) {

for(unsigned int j = 0; j < miscya\_in\_ryad; j++) {

if(!miscya[i][j]) {

count++;

}

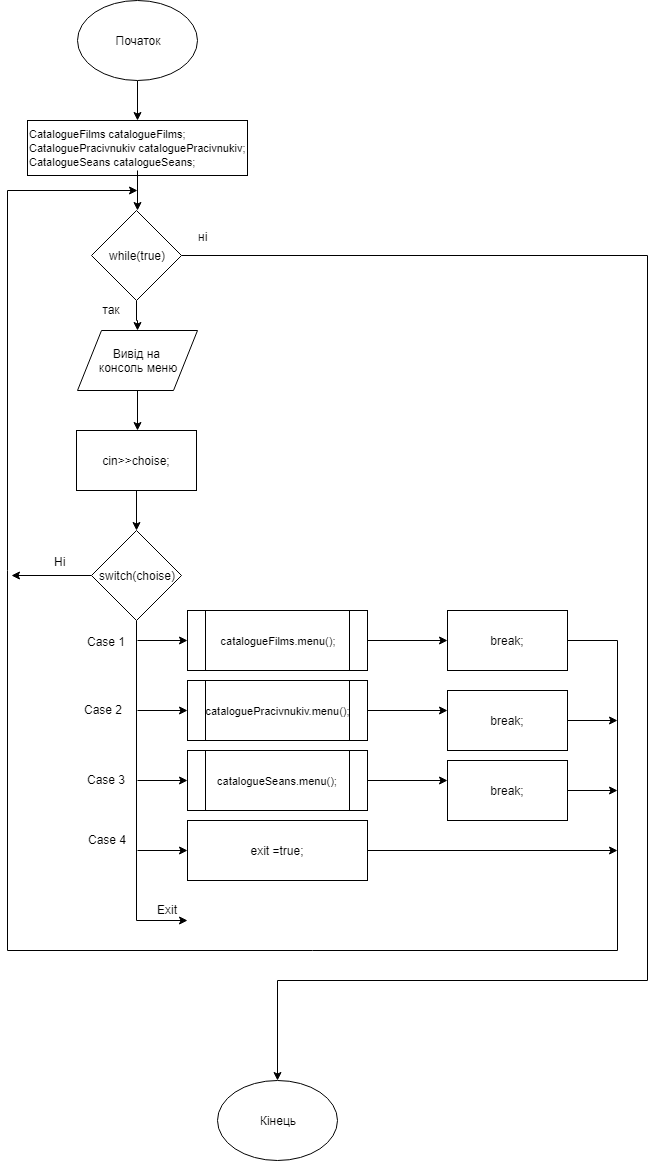
}

}

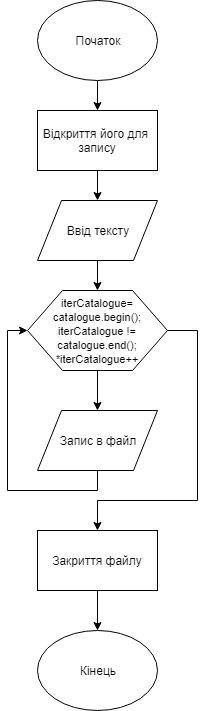
return count;

}

**Додаток Б**

Блок схема головної функції int main():

Блок-схема для запису файлів у файл:



Блок-схема для методу void CatalogueSeans::profit():

