

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего
образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ
И ПРОГРАММНОЙ ИНЖЕНЕРИИ (КАФЕДРА №43)

ОТЧЕТ ЗАЩИЩЕН С ОЦЕНКОЙ: _____

ПРЕПОДАВАТЕЛЬ:

<u>Старший преподаватель</u>	_____	<u>Е. В. Павлов</u>
(должность, уч. степень, звание)	(подпись, дата)	(инициалы, фамилия)

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №1

«СОСТАВЛЕНИЕ ВАРИАНТОВ ТЕСТИРОВАНИЯ
И МЕТРИКА ПОКРЫТИЯ ТРЕБОВАНИЙ»

ПО КУРСУ: «МЕТРОЛОГИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ»

РАБОТУ ВЫПОЛНИЛ СТУДЕНТ:	<u>4033</u>	/	<u>Х.В.Сидиропуло</u>
	(номер группы)		(инициалы, фамилия)

/ _____ /	<u>18.03.2022</u>
(подпись студента)	(дата отчета)

Санкт-Петербург 2022

1. Цель работы

Целью данной работы является изучение артефактов ручного тестирования функций приложения и оценка плотности покрытия тестами функциональных требований.

2. Задание на лабораторную работу

В соответствии с индивидуальным вариантом задания, сформулированными бизнес-правилами и представленными функциональными требованиями, необходимо разработать приложение, отвечающее заданным ограничениям на реализацию.

Составить не менее десяти тест кейсов для тестирования функций данного приложения (при описании тест кейсов на каждое действие пользователя должна быть ответная реакция системы).

Построить матрицу соответствия требований для анализа тестового покрытия.

Выполнить тестирование приложения по составленным тест кейсам.

Вариант задания - 87
Приложение для поиска кинотеатров и просмотра расписания сеансов.

3. Спецификация требований к программному обеспечению

Данное приложение представляет собой адаптацию под сформулированную концепцию продукта и предназначено для просмотра фильмов. Основные функции приложения заключаются в предоставлении доступа к расписанию сеансов.

Функциональные требования к приложению представлены в таблице 1 (каждому требованию соответствует уникальный идентификатор):

Таблица 1 — Функциональные требования

FR-01	Сохранение даты и времени запуска приложения
FR-02	Добавление записи (формат записи: номер сеанса, время начала, длительность фильма, название, цена, ограничение по возрасту. формат кино (2D,3D и тп), номер зала
FR-02.1	Добавление стандартного списка фильмов

FR-03	Проверка записи на корректность (Запись должна состоять из целых чисел без использования символов. в некоторых случаях не превышать определенные лимиты)
FR-04	Сортировка фильмов по номеру сеанса в файле
FR-05	Редактировать определенную запись
FR-06	Вывести записи в консоль
FR-06.1	Вывести отдельные записи по критериям
FR-07	Удалить определенную запись
FR-08	Удалить все записи
FR-08.1	В связи с условиями пользования, существует функция удаления exe файла выдаваемой копии приложения (вызов отдельного bat файла) (защищена паролями)

Таблица 2 - Нефункциональные требования

NFR-01	Приложение должно использовать текстовый интерфейс пользователя
NFR-02	Хранение любой релевантной информации должно осуществляться посредством внешних файлов
NFR-03	После выполнения действия пользователя (выбор пункта меню) и перед выводом информации, должна производиться очистка экрана консоли
NFR-04	Приложение должно обеспечить возможность вернуться к предыдущему пункту меню (или в главное меню)

Таблица 3 — Ограничения реализации

DIR-01	Программный код управляющей логики приложения написан на языке C++
--------	--

4. Тест дизайн и анализ тестового покрытия

ID и наименование: TC-1 Сохранение даты и времени запуска приложения	
Ссылка на FR-01 требование:	
Дата создания: 18.03.2022	
Автор: Хетаг	
Предусловия: предварительная настройка среды или системы не требуется	
Шаги теста:	Ожидаемый результат:
1. Запустить приложение	1. Дата и время запуска приложения будут записаны time.txt 2. Приложение выведет сообщение о записи текущей даты.

Результат :



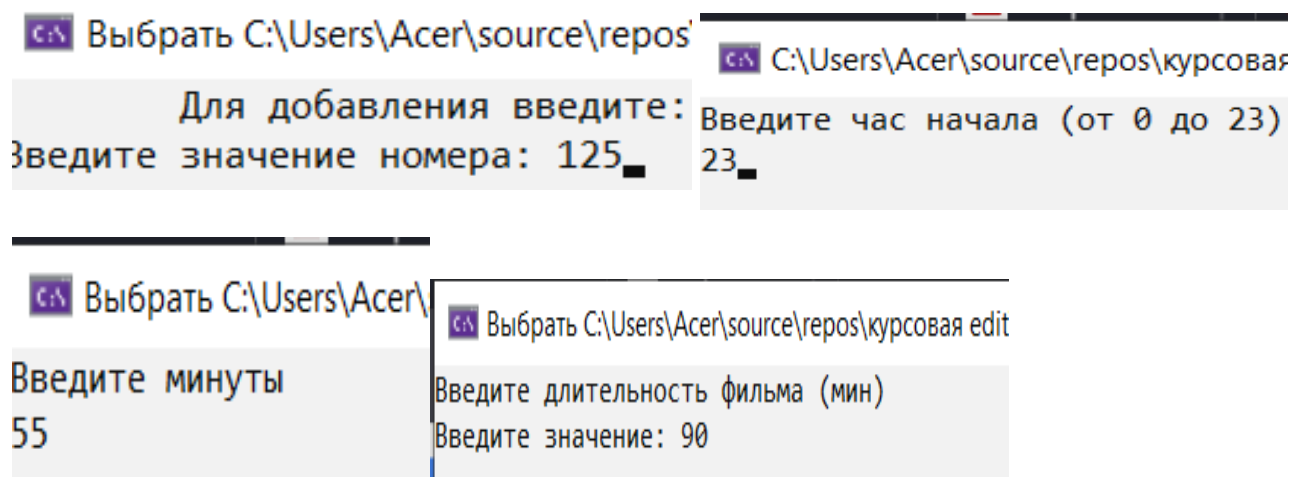
Содержание файла time.txt :



time – Блокнот

Файл	Правка	Формат	Вид	Справка
18/3/2022		3:38:58		
18/3/2022		3:39:26		
18/3/2022		3:42:4		
18/3/2022		3:42:33		
18/3/2022		3:44:4		
18/3/2022		3:44:50		
18/3/2022		3:45:7		

ID и наименование: TC-2 Добавление сеанса фильма	
Ссылка на требование: FR-02	
Дата создания: 18.03.2022	
Автор: Хетаг	
Предусловия: предварительная настройка среды или системы не требуется	
Шаги теста:	Ожидаемый результат:
<p>1. Выбрать в главном меню пункт:</p> <p>«1 – Добавление записи»;</p> <p>2. Ввести номер сеанса, время начала фильма, длительность фильма, цену билета, возрастное ограничение, формат фильма, номер зала</p>	<p>1. Приложение выводит список действий</p> <p>2. Приложение просит ввести данные по очереди</p> <p>3. После каждого ввода консоль очищается</p> <p>4. Запись добавлена в файл</p>



Выбрать C:\Users\Acer\source\repos\	C:\Users\Acer\source\repos\
Название фильма: Брат	Введите цену Введите значение: 500

Выбрать C:\Users\Acer\source\repos\
Введите возрастное ограничение Введите значение: 18

Выбрать C:\Users\Acer\source\repos\	Выбрать C:\Users\Acer\source\repos\
Введите формат кино [2-3...][D] Введите значение: 3	Введите номер зала Введите значение: 7

C:\Users\Acer\source\repos\курсовая edit\
Запись занесена Добавление в файл завершено
Меню
0 - создание списка по умолчанию
1 - добавление записи
2 - редактирование записи
3 - удаление записи
4 - поиск фильма
5 - сортировка
6 - вывод на экран
7 - загрузка данных из файла
8 - сохранение данных в файл

ID и наименование: TC-2.1 Добавление сеансов	
Ссылка на требование: FR-03.1	
Дата создания: 18.03.2022	
Автор: Хетаг	
Предусловия: предварительная настройка среды или системы не требуется	
Шаги теста: 1) Выбрать создание списка по умолчанию	Ожидаемый результат: 1) Приложение добавит стандартный список прописанный в коде 2) Выводится сообщение о загрузке данных в файл и занесение каждой записи


Запись занесена
Запись занесена
Запись занесена
Запись занесена
Добавление в файл завершено

text – Блокнот

Файл	Правка	Формат	Вид	Справка
------	--------	--------	-----	---------

133,Брат-2;11:20|190|700|18|3|10|
331,Брат-1;13:20|108|250|18|3|7|
227,Защитники;17:20|720|100|7|2|6|
228,Кухня 2;17:25|255|200|18|2|4|
125,Елки-19;18:30|240|500|18|3|13|

ID и наименование: TC-3 Проверка на корректность ввода	
Ссылка на требование: FR-03	
Дата создания: 18.03.2022	
Автор: Хетаг	
Предусловия: предварительная настройка среды или системы не требуется	
Шаги теста:	Ожидаемый результат:
1. Выбрать в главном меню пункт: «1 – Добавление записи»; 2. Ввести номер сеанса буквами 3. Дойти до ввода времени 4. Ввести 24 5. Ввести в длительность фильма отрицательное значение	1. Приложение выводит добавление номера сеанса 2. Приложение просит ввести данные 3. После каждого ввода консоль очищается 4. Приложение выводит сообщение об ошибке ввода и предлагает ввести заново 5. Приложение выводит сообщение об ошибке ввода и предлагает ввести заново 6. Приложение выводит сообщение об ошибке ввода и предлагает ввести заново


 Выбрать C:\Users\Acer\source\repos\

Введите час начала (от 0 до 23)


Для добавления введите:

Введите значение номера: мпо 24

Неверный ввод

Ошибка ввода

Введите значение номера: Введите час начала (от 0 до 23)


 Выбрать C:\Users\Acer\source\repos\кур

Введите длительность фильма (мин)

Введите значение: -100

Ошибка ввода

Введите значение:

ID и наименование:	ТС-4 Сортировка записей		
Ссылка на требование:	FR-04		
Дата создания:	18.03.2022		
Автор:	Хетаг		
Предусловия: предварительная настройка среды или системы не требуется			
Шаги теста:		Ожидаемый результат:	
1)Выбрать сортировку		1)Приложение сортирует записи в файле по номеру сеанса 2) Выводится сообщение о том, что файл отсортирован, после чего выводится отсортированный список	

Добавление в файл завершено
Список отсортирован
расписание фильмов
Всего фильмов: 5

№ сеанса	Название фильма	время начала	длительность(мин)	цена билета	Возраст	формат	номер зала
125	Елки-19	18:30	190	700Р	18	3D	10
133	Брат-2	11:20	108	250Р	18	3D	7
227	Защитники	17:20	720	100Р	7	2D	6
228	Кухня 2	17:25	255	200Р	18	2D	4
331	Брат-1	13:20	240	500Р	18	3D	13

ID и наименование: TC-5 Редактировать определенную запись

Ссылка на требование: FR-05

Дата создания: 18.03.2022

Автор: Хетаг

Предусловия: предварительная настройка среды или системы не требуется

Шаги теста:

- 1)Выбрать редактирование
- 2)Выбрать параметр для изменения
- 3) Ввести новое значение

Ожидаемый результат:

- 1)Приложение просит ввести номер сеанса для редактирования
- 2) Приложение просит ввести номер параметра для изменения
- 3) Сообщение о добавлении в файл

№ сеанса	Название фильма	время начала	длительность(мин)	цена билета	Возраст	формат	номер зала
133	Брат-2	11:20	190	700Р	18	3D	10

Введите номер сеанса для изменения параметров: 133

Введите номер для изменения информации

1. Номер сеанса
2. Название фильма
3. Время начала
4. Длительность
5. Цена
6. Ограничение
7. Новый формат
- 8.Зал 0. Выход

Ваш выбор:

Введите номер для изменения информации

1. Номер сеанса
2. Название фильма
3. Время начала
4. Длительность
5. Цена
6. Ограничение
7. Новый формат
- 8.Зал 0. Выход

Ваш выбор: 2

Новое название:

Название фильма: Джокер

Добавление в файл завершено

№ сеанса	Название фильма	время начала	длительность(мин)	цена билета	Возраст	формат	номер зала
133	Джокер	11:20	190	700Р	18	3D	10

ID и наименование: TC-6 Вывод записей в консоли	
Ссылка на требование: FR-06	
Дата создания: 18.03.2022	
Автор: Хетаг	
Предусловия: предварительная настройка среды или системы не требуется	
Шаги теста:	Ожидаемый результат:
1)Выбрать вывод на экран	1)Приложение выводит расписание фильмов и их количество 2) Приложение выводит меню для дальнейших действий

расписание фильмов

Всего фильмов: 5

№ сеанса	Название фильма	время начала	длительность(мин)	цена билета	Возраст	формат	номер зала
133	Джокер	11:20	190	700P	18	3D	10
331	Брат-1	13:20	108	250P	18	3D	7
227	Защитники	17:20	720	100P	7	2D	6
228	Кухня 2	17:25	255	200P	18	2D	4
125	Елки-19	18:30	240	500P	18	3D	13

Меню

- 0 - создание списка по умолчанию
- 1 - добавление записи
- 2 - редактирование записи
- 3 - удаление записи
- 4 - поиск фильма
- 5 - сортировка
- 6 - вывод на экран

ID и наименование: TC-6.1 Вывод отдельной записи по критериям	
Ссылка на требование: FR-06	
Дата создания: 18.03.2022	
Автор: Хетаг	
Предусловия: предварительная настройка среды или системы не требуется	
Шаги теста: 1)Выбрать поиск фильма 2) Ввести критерий поиска	Ожидаемый результат: 1)Приложение просит ввести номер критерия поиска 2) Приложение выводит фильм по критерию

Введите чтобы продолжить поиск
1 - поиск по номеру
2 - поиск по названию
3 - поиск по времени сеанса
0 - Выход
2_

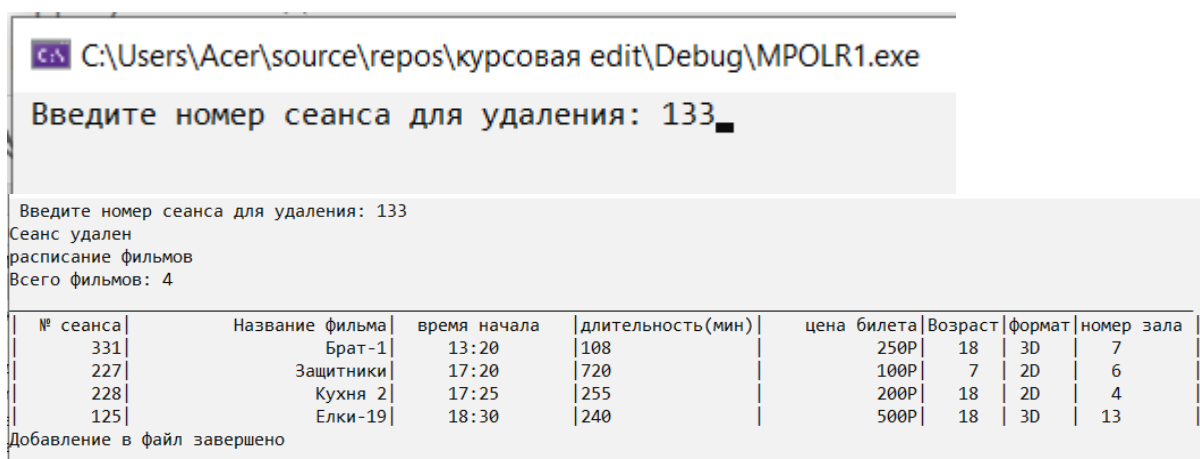
Введите чтобы продолжить поиск
1 - поиск по номеру
2 - поиск по названию
3 - поиск по времени сеанса
0 - Выход
2

Введите название: Брат-2
расписание фильмов
Всего фильмов: 5

№ сеанса	Название фильма	время начала
133	Брат-2	11:20

Меню
0 - создание списка по умолчанию
1 - добавление записи


ID и наименование: TC-7 Удаление определенной записи	
Ссылка на требование: FR-07	
Дата создания: 18.03.2022	
Автор: Хетаг	
Предусловия: предварительная настройка среды или системы не требуется	
Шаги теста: 1)Выбрать удаление записи 2) Ввести номер сеанса для удаления	Ожидаемый результат: 1)Приложение просит ввести номер сеанса для удаления 2) Приложение выводит расписание сеансов и обновляет файл



ID и наименование: TC-8 Удалить все записи	
Ссылка на требование: FR-08	
Дата создания: 18.03.2022	
Автор: Хетаг	
Предусловия: предварительная настройка среды или системы не требуется	
Шаги теста: 1)Выбрать удалить все	Ожидаемый результат: 1)Приложение очищает консоль и файл 2) Приложение выводит сообщения о том, что сеансы удалены, бд(файл) удалена и список сеансов пуст

<pre> Меню 0 - создание списка по умолчанию 1 - добавление записи 2 - редактирование записи 3 - удаление записи 4 - поиск фильма 5 - сортировка 6 - вывод на экран 7 - загрузка данных из файла 8 - сохранение данных в файл 9 - удалить все 10- exit 11 - staff only Ваш выбор: 9_ </pre>	<pre> C:\>Выбрать C:\Users\Acer\source' Сеансы удалены бд удалена Список пуст </pre>
--	---

ID и наименование: TC-8.1 Удалить все файлы	
Ссылка на требование: FR-08	
Дата создания: 18.03.2022	
Автор: Хетаг	
<p>Предусловия: требуется предварительная установка bat файла</p> <p>// комментарий</p> <p>Эта функция создана для полного удаления файлов граждан, чья лицензия на использование данного ПО (4 недели) истекла. Функция находится под паролями, которые известны только сотрудникам роскомнадзора, для ужаса применено злостное оформление консоли.</p>	
<p>Шаги теста:</p> <ol style="list-style-type: none"> 1)Выбрать stuff only 2) Ввести пароль 3) Ввести второй пароль 4) Если хоть один пароль не подошел - бежать. 5) Если пароли подошли, то можно ещё 4 недели пользоваться приложением РУСКИНОПОИСК <p>(ни один студент и ни одна папка ЛР не пострадала при запуске данной функции, все проводилось в тестовом режиме, не повторяйте дома)</p>	<p>Ожидаемый результат:</p> <ol style="list-style-type: none"> 1)Приложение запрашивает пароль 2) Приложение сверяет пароль, если он верный, то просит второй пароль от сотрудника, который получает этот пароль на госуслугах, если же первый пароль не верный, то смотреть пункт 3. 3) Если пароль не верный, запускается bat файл, который удаляет папку(в случае гражданина программу.exe) и бьет тревогу. Все что осталось от папки это пара КБ лог файлов для поимки экстремиста. 4) Подтверждение для удаления роскомнадзор обещал убрать в следующем обновлении. 5) Если же оба пароли совпали, то лицензия будет продлена.

 C:\Users\Acer\source\repos\курсовая edit\Debug\MPOLR1.exe

Введите пароль сотрудника роскомнадзора

ROSCOMNADZOR

Лицензия пользования ПО истекла

Введите пароль администратора с госуслуг для продления лицензии:

Введите пароль сотрудника роскомнадзора

ROSCOMNADZOR

Лицензия пользования ПО истекла

Введите пароль администратора с госуслуг для продления лицензии:

hackroscmnadzor

Попытка взлома, за вами уже выехали, отряд омона подъедет к вам через 3 минуты, ожидайте

При попытке сбежать натравим на тебя росгвардию

Самоуничтожение программы запущено...

C:\Users\Acer\source\repos\курсовая edit>ping -n 1 -w 1000 0.0.0.0 1>nul

C:\Users\Acer\source\repos\курсовая edit>del C:\Users\Acer\source\repos\test

C:\Users\Acer\source\repos\test*, вы уверены [Y(да)/N(нет)]? _

Лицензия пользования ПО истекла

Введите пароль администратора с госуслуг для продления лицензии:

hackroscmnadzor

Попытка взлома, за вами уже выехали, отряд омона подъедет к вам через 3 минуты, ожидайте

При попытке сбежать натравим на тебя росгвардию

Самоуничтожение программы запущено...

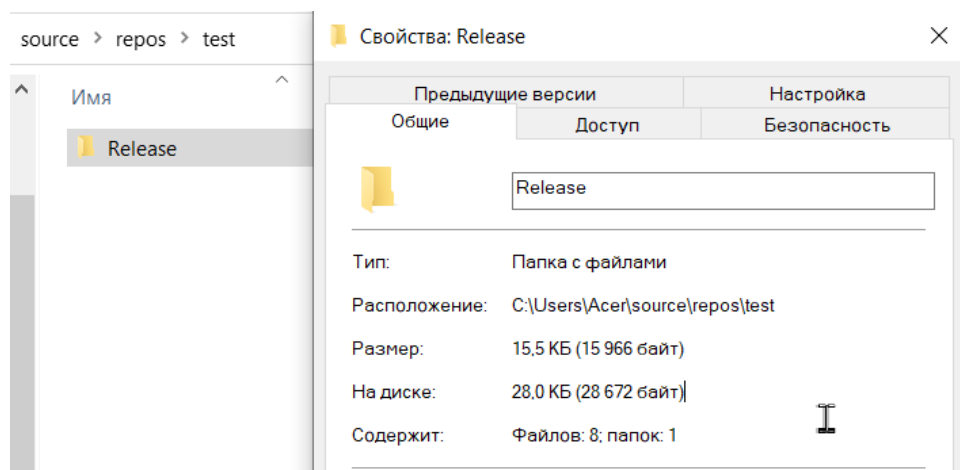
C:\Users\Acer\source\repos\курсовая edit>ping -n 1 -w 1000 0.0.0.0 1>nul

C:\Users\Acer\source\repos\курсовая edit>del C:\Users\Acer\source\repos\test

C:\Users\Acer\source\repos\test*, вы уверены [Y(да)/N(нет)]? Y

C:\Users\Acer\source\repos\курсовая edit>del C:\Users\Acer\source\repos\test\Release

C:\Users\Acer\source\repos\test\Release*, вы уверены [Y(да)/N(нет)]? Y



Введите пароль сотрудника роскомнадзора

ROSCOMNADZOR

Лицензия пользования ПО истекла

Введите пароль администратора с госуслуг для продления лицензии:

SIDIROPULO4033

Ладно, оплатите налог

Выполним анализ тестового покрытия при помощи построения матрицы соответствия требований:

Требование	FR 1	FR 2	FR 2.1	FR 3	FR 4	FR 5	FR 6	FR 6.1	FR 7	FR 8	FR8.1
тест кейсы	1	2	2	1	1	1	1	1	1	2	2
ТС - 1	✓										
ТС – 2		✓									
ТС – 2.1			✓								
ТС – 3				✓							
ТС - 4					✓						
ТС – 5						✓					
ТС – 6							✓				
ТС – 6.1								✓			
ТС - 7									✓		
ТС - 8										✓	
ТС- 8.1											✓

Таким образом, из данной матрицы следует, что для всех функциональных требований приложения есть хотя бы по одному тесту, соответственно тестовое покрытие функциональных требований в данном случае составляет 100%.

Выводы по работе

В результате выполнения данной работы было разработано консольное приложение для расписания киносеансов. Составлено необходимое количество тест кейсов, в соответствии с которыми выполнено тестирование приложения, и построена матрица соответствия требований для анализа тестового покрытия.

Анализ покрытия показывает, что составленные тесты покрывают 100% функциональных требований, однако нельзя говорить о том, что само приложение было полностью протестировано, так как выбранная метрика тестового покрытия работает только с требованиями и не учитывает конечную реализацию,

Также в приложении реализована защита от «дурака» посредством следующих проверок:

- Для ввода любых целочисленных переменных предусмотрена проверка пропускающая только положительные численные значения.
- При неверном вводе искомого номера для определенных целей пользователю предлагается либо продолжить, либо вызвать функцию заново.
- Предусмотрен номер сеанса как критерий, который находится всегда уникальным, проверка на наличие номера присутствует и не получится сделать полный дубликат.

Это не гарантия полной работоспособности программе при выпуске

- Нет контроля дублирующей информации (номера сеанса) при загрузке из файла, если его кто-то поменяет вручную.

Таким образом, можно заключить, что выполненная работа соответствует поставленной задаче и отвечает всем сформулированным в методических указаниях требованиям.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Павлов Е. В. Методические рекомендации к выполнению лабораторных работ: Метрология программного обеспечения / Евгений Васильевич Павлов. — СПб ГУАП, 2022
- 2 . Павловская, Т.А. С/ С++. Программирование на языке высокого уровня / Т.А. Павловская. – СПб.: Питер, 2011 – 461 с.
3. Куликов, С. С. Тестирование программного обеспечения. Базовый курс / С. С. Куликов — Минск: Четыре четверти, 2017 — 312 с.
4. Кент Бек. Экстремальное программирование. Разработка через тестирование

Исходный код программы

Batnik.bat

```
ping -n 1 -w 1000 0.0.0.0 >nul  
del C:\Users\Acer\source\repos\test
```

```
del C:\Users\Acer\source\repos\test\Release
```

Header.h

```
#pragma once  
#define _CRT_SECURE_NO_WARNINGS  
#include <iostream>  
  
#include <windows.h>  
  
#include <string>  
  
#include <fstream>  
  
#include <iomanip>  
#include <ctime>  
#include <random>  
  
using std::cout;  
using std::endl;  
using std::cin;  
using std::getline;  
using std::string;  
  
class TableList;  
  
class Cinema {  
private:  
    string title;    // название  
    string time;     // время начала  
    int timeEnd;    // длительность фильма  
    int number;     // номер сеанса  
    int price;      // цена  
    int age;        // ограничение по возрасту  
    int type;       // формат  
    int room;       // номер зала  
    Cinema* next = NULL;  
  
public:  
    Cinema(string, string, int, int, int, int, int, int);  
    friend TableList;  
};  
  
class TableList {  
private:  
    Cinema* first = NULL;
```

```

        Cinema* current = NULL;

public:

    void add(string, string, int, int, int, int, int,int); // добавление элемента

    void show(); // вывод

    int checkvalue(); // проверка значений на int

    void remove_elem(); // удаление элемента
    void deleteall();
    void remove_database(); // очистка файла

    void search(); // поиск

    void info(); // вывод шапки таблицы

    int get_count(); // количество элементов

    void edit(); // редактирование

    void sort(); // сортировка

    void swap(Cinema*&, Cinema*&); // поменять местами

    void save(); //функция для сохранения в файл

    void load(); //функция для загрузки из файла

    int check_number(); // проверки номера

    string checkName(); // проверка названия

    unsigned short int checkMinutes(); //проверка минут

    unsigned short int checkHours(); // проверка часов
};

```

Cinema.cpp

```

#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <windows.h>
#include <iomanip>
#include <string>
#include <fstream>
#include "Header.h"
#include <type_traits>
#include <stdio.h>
#include <stdlib.h>
#include <ctime>
#include <time.h>
using std::cin;
using std::cout;
using std::endl;

```



```

using std::setw;
using std::getline;
using std::to_string;
using std::get;
using std::ios;

```

```

using std::string;
using std::fstream;
using std::ofstream;
using std::ifstream;

```

```

Cinema::Cinema(string title, string time, int timeEnd, int number, int price, int age, int type,
int room ) {

```

```

    this->title = title;
    this->time = time;
    this->number = number;
    this->price = price;
    this->timeEnd = timeEnd;
    this->age = age;
    this->type = type;
    this->room = room;
}

```

```

// вывод шапки таблицы

```

```

void TableList::info() {

```

```

    cout << "расписание фильмов" << endl;
    cout << "Всего фильмов: " << get_count() << endl;
    cout <<
    "
    _____" << endl;
    cout << "|" << setw(10) << "№ сеанса" << "|" << setw(25) << "Название фильма"
    << "|" << setw(10) << " время начала " << setw(1) << "|" << "длительность(мин)"<< "|" <<
    setw(15) << "цена билета" << "|" << "Возраст"<< "|" << setw(5) << "формат" << "|"
    << setw(5) << "номер зала" << setw(1) << "|" << endl;
}

```

```

// добавление элемента

```

```

void TableList::add(string title, string time, int timeEnd, int number, int price, int age, int type,
int room) {

```

```

    if (this->first == NULL) {

```

```

        Cinema* film = new Cinema(title, time, timeEnd, number, price, age, type,
room);
        this->first = film;
        this->current = film;

    }

    else {
        Cinema* temp_current = this->first;
        while (temp_current->next != NULL)
            temp_current = temp_current->next;
        Cinema* film = new Cinema(title, time, timeEnd, number, price, age, type,
room);
        this->current = film;
        temp_current->next = film;

    }

    cout << "Запись занесена " << endl;

}

// вывод

void TableList::show() {

    if (this->first == NULL)

        cout << "Список пуст " << endl;

    else {

        info();

        Cinema* temp_current = this->first;

        while (temp_current != NULL) {
            // temp_current->price
            cout << "|" << setw(10) << temp_current->number << "|" << setw(25)
<< temp_current->title << "|" << setw(10) << temp_current->time << "    " << temp_current-
>timeEnd << setw(15) << "|" << setw(14) << temp_current->price << "P|" << setw(5) <<
temp_current->age << setw(3) << "|" << setw(2) << temp_current->type << "D" << setw(4) <<
"|" << setw(4) << temp_current->room << setw(8) << "|" << endl;

            temp_current = temp_current->next;

        }

    }
}

```

```

    }

}

// удаление бд

void TableList::remove_database() {

    ofstream in("text.txt", ios::trunc);
    in.close();
    cout << "бд удалена" << endl;
}

// удаление элемента
void TableList::remove_elem() {
    int num;
    int k = 0;
    Cinema* temp_current = this->first;

    if (temp_current != nullptr) {
        Cinema* temp_second = temp_current->next;
    }

    if (temp_current != nullptr or NULL || first != nullptr or NULL)
    {
        cout << " Введите номер сеанса для удаления: ";
        cin >> num;
        while (num < 100 || num > 999 || cin.fail()) { // проверка на
трехзначный номер поезда
            cout << " Неверное значение, введите ещё раз: " <<
endl;

            cin.clear();
            cin.ignore(cin.rdbuf()->in_avail());

        }

        if (temp_current->number == num) {

            Cinema* temp = first;
            first = first->next;
            delete temp;
            k++;
            cout << "Сеанс удален" << endl;
        }
        else
        {
            while (temp_current->next != NULL)
            {
                if (temp_current->next->number == num)

```

```

        {
            break;
        }
        temp_current = temp_current->next;
    }
    if (temp_current->next != nullptr)
    {
        Cinema* deltrain = temp_current->next;
        temp_current->next = temp_current->next->next;
        delete deltrain;
        k++;
    }
}

if (k == 0) {
    cout << "\tНомера нет" << endl;
    int x;
    cout << "1- Начать удаление заново" << endl;
    cout << "0 - Выход в меню" << endl;
    cout << "Ваш выбор: ";

    while (true) {

        cin >> x;

        if (x < 0 || x > 1 || cin.fail())
        {

            cout << " Повторите ввод\n";
            cout << "Ваш выбор: ";
            cin.ignore(32767, '\n');
            cin.clear();
            cin.ignore(cin.rdbuf()->in_avail());
        }
        else {
            break;
        }
    }
    switch (x)
    {
    case 0:
        k=0;
        break;
    case 1:
        k=0;
        remove_elem();
    }
}

```

```

        }
    }
}

// удаление всего списка
void TableList::deleteall() {

    Cinema* temp_current = this->first;

    while (temp_current-> number != NULL and first!= NULL)
    {

        Cinema* temp = first;

        first = first->next;
        delete temp;

    }

    cout << "Сеансы удалены" << endl;
}

// поиск

void TableList::search() {

    Cinema* temp_current = this->first;

    int choose_search, num;

    string word;

    if (temp_current != NULL) {

        cout << "Введите чтобы продолжить поиск " << endl;

        cout << "1 - поиск по номеру" << endl;

        cout << "2 - поиск по названию" << endl;
    }
}

```

```

cout << "3 - поиск по времени сеанса " << endl;

cout << "0 - Выход" << endl;

while (true) { // цикл продолжается до тех пор, пока пользователь не
введет корректное значение

    cin >> choose_search;

    if (choose_search < 0 || choose_search > 3 || cin.fail()) {

        cout << " ----- " << endl;

        cout << "| Неверное значение, введите ещё раз | \n";

        cout << " ----- " << endl;

        cin.ignore(32767, '\n');
        cin.clear();
        cin.ignore(cin.rdbuf()->in_avail());

    }

    else

        break;

}

switch (choose_search)

{

case 1:

    cout << "Введите номер: ";

    cin >> num;

    info();

    while (temp_current != NULL) {

        if (temp_current->number == num) {

            cout << "|" << setw(10) << temp_current->number <<
"|" << setw(25) << temp_current->title << "|" << setw(10) << temp_current->time << "    |"

```

```
<< temp_current->timeEnd << setw(11) << "|" << setw(14) << temp_current->price << "P|"
<< setw(5) << temp_current->age << setw(3) << "|" << setw(2) << temp_current->type <<
"D" << setw(4) << "|" << setw(4) << temp_current->room << setw(8) << "|" << endl;
```

```
break;
```

```
}
```

```
temp_current = temp_current->next;
```

```
}
```

```
break;
```

```
case 2:
```

```
cout << "Введите название: ";
```

```
cin >> word;
```

```
info();
```

```
while (temp_current != NULL) {
```

```
    if (temp_current->title == word)
```

```
        cout << "|" << setw(10) << temp_current->number <<
        "|" << setw(25) << temp_current->title << "|" << setw(10) << temp_current->time << "    |"
        << temp_current->timeEnd << setw(11) << "|" << setw(14) << temp_current->price << "P|"
        << setw(5) << temp_current->age << setw(3) << "|" << setw(2) << temp_current->type <<
        "D" << setw(4) << "|" << setw(4) << temp_current->room << setw(8) << "|" << endl;
```

```
        temp_current = temp_current->next;
```

```
    }
```

```
break;
```

```
case 3:
```

```
cout << "Введите время: ";
```

```
cin >> word;
```

```
info();
```

```
while (temp_current != NULL) {
```

```
    if (temp_current->time == word)
```

```

        cout << "|" << setw(10) << temp_current->number <<
        "|" << setw(25) << temp_current->title << "|" << setw(10) << temp_current->time << "    |"
        << temp_current->timeEnd << setw(11) << "|" << setw(14) << temp_current->price << "P|"
        << setw(5) << temp_current->age << setw(3) << "|" << setw(2) << temp_current->type <<
        "D" << setw(4) << "|" << setw(4) << temp_current->room << setw(8) << "|" << endl;
        temp_current = temp_current->next;
    }

    break;

default:
    break;

}

}

else
    cout << "Список пуст " << endl;
}

```

```

void TableList::edit() {
    int k = 0;
    Cinema* temp_current = this->first;
    if (first == nullptr) {
        cout << "Список пуст " << endl;
    }
    int chose;
    int number;
    if (first != nullptr or temp_current != NULL)
    {
        cout << "Введите номер сеанса для изменения параметров: ";
        cin >> number;
        while (number < 100 || number > 999 || cin.fail())
        { // проверка на трехзначный номер поезда

            cout << " Неверное значение, введите ещё раз: ";
            cin.ignore(32767, '\n');
            cin.clear();

        }
    }
    unsigned short int min, hour;

    string s_h, s_m;
}

```



```

if (temp_current != NULL) {

    while (temp_current != NULL)

    {

        int num = temp_current->number;
        if (number == num) {
            k++;
            cout << "Введите номер для изменения информации" <<
endl;

            cout << "1. Номер сеанса\n 2. Название фильма\n 3. Время
начала\n 4. Длительность\n 5. Цена \n 6. Ограничение \n 7. Новый формат\n 8.Зал 0.
Выход " << endl << "Ваш выбор: ";

            cin >> chose;

            int new_number; string new_title, new_time;
            int new_price; int new_timeend; int new_room; int new_type;
int new_age;

            switch (chose) {

                case 1:

                    cout << "Новый номер сеанса: ";

                    cin >> new_number;

                    temp_current->number = new_number;

                    break;

                case 2:

                    cout << "Новое название: " << endl;

                    new_title = checkName();

                    temp_current->title = new_title;

                    break;
                    // длительность цена возраст формат номер зала
                case 4:

                    cout << "Новая длительность фильма" << endl;
                    new_timeend = checkvalue();

```

```

temp_current->timeEnd = new_timeend;
break;

case 5:
    cout << "Введите новую цену" << endl;
    new_price = checkvalue();
    temp_current->price = new_price;
    break;

case 6:
    cout << "Введите новое возрастное ограничение" <<

endl;

    new_age = checkvalue();
    temp_current->age = new_age;
    break;

case 7:
    cout << "Введите новый формат фильма" << endl;
    new_type = checkvalue();
    temp_current->type = new_type;
    break;

case 8:
    cout << "Введите новый номер зала" << endl;
    new_room = checkvalue();
    temp_current->room = new_room;
    break;

case 3:

    cout << "Новое время начала: " << endl;

    hour = checkHours();

    if (hour < 10)

        s_h = "0" + to_string(hour);

    else

        s_h = to_string(hour);

    min = checkMinutes();

    if (min < 10)

        s_m = "0" + to_string(min);

    else

```

```

        s_m = to_string(min);

        new_time = s_h + ":" + s_m;

        temp_current->time = new_time;

        break;

    case 0:

        break;

    default:

        cout << " че стало???" << endl;

        break;

    }

}

temp_current = temp_current->next;

}

}

if (k == 0) {
    cout << "\tНомера нет" << endl;
    int x;
    cout << "1- Начать редактировать заново" << endl;
    cout << "0 - Выход в меню" << endl;
    cout << "Ваш выбор: ";

    while (true) {

        cin >> x;

        if (x < 0 || x > 1 || cin.fail())
        {

            cout << " Повторите ввод\n";
            cout << "Ваш выбор: ";
            cin.ignore(32767, '\n');
            cin.clear();
            cin.ignore(cin.rdbuf()->in_avail());

        }
    }
}

```

```

        else {
            break;
        }
    }
    switch (x)
    {
    case 0:
        k = 0;
        break;
    case 1:
        k = 0;
        edit();
    }
}
}

```

```

int TableList::get_count() {

    Cinema* temp_current = this->first;

    int k = 0;

    while (temp_current != NULL) {

        k++;

        temp_current = temp_current->next;
    }
    return k;
}

```

```

void TableList::swap(Cinema*& one, Cinema*& two) {

    string title = one->title;

    string time = one->time;

    int number = one->number;

    one->title = two->title;

    one->time = two->time;
}

```

```

        one->number = two->number;

        two->title = title;

        two->time = time;

        two->number = number;
    }

void TableList::sort() {

    bool swaps = 1;

    Cinema* head = first;

    Cinema* tmp1 = head;

    while (swaps) {

        swaps = false;

        tmp1 = head;

        while (tmp1 != NULL && tmp1->next != NULL) {

            if (tmp1->number > tmp1->next->number) {

                swap(tmp1, tmp1->next);

                swaps = true;

            }

            tmp1 = tmp1->next;

        }

        save();

    }

}

//функция для сохранения в файл

void TableList::save() {

```

```

Cinema* temp = first;

if (temp != NULL) {

    ofstream fout("text.txt");

    if (!fout.is_open())

        cout << "Файл не может быть открыт!\n";

    else {

        while (temp != NULL) {

            fout << temp->number << "," << temp->title << ";" << temp-
>time << "|" << temp->timeEnd << "|" << temp->price << "|" << temp->age << "|" << temp-
>type << "|" << temp->room << "|" << endl;

            temp = temp->next;

        }

        cout << "Добавление в файл завершено\n";

    }

    fout.close();

}

else {

    cout << "Список пуст\n";

}

}

```

//функция для загрузки из файла

```

void TableList::load()
{
    int check = 0;

    char line[250];
    Cinema* temp = first;
    ifstream fin("text.txt");

```

```

string time;
string num;
string str;
string title;
string money;
string year;
string filmtype;
string roomnum;
string timeend;
int times = 0;
int price = 0;
int x = 0;
int i = 0;
int number = 0;
int room = 0;
int type = 0;
int age = 0;

bool check_to_null = false;
do {

    fin.getline(line, 250);

    if (line[0] != '\0') {

        for (int i = 0; i < 50; i++)
        {
            str += line[i];
        }

        x = 0;

        while (str[x] != ',')
        {

            num += str[x];
            x++;
            number = atoi(num.c_str());

        }

        x++;
        while (str[x] != ';')
        {

            title += str[x];
            x++;

```

```

}

x++;
while (str[x] != '|')
{
    time += str[x];
    x++;

}
x++;
while (str[x] != '|')
{
    timeend += str[x];
    x++;
    times = atoi(timeend.c_str());

}
x++;
while (str[x] != '|')
{

    money += str[x];
    x++;
    price = atoi(money.c_str());

}
x++;
while (str[x] != '|')
{

    year += str[x];
    x++;
    age = atoi(year.c_str());

}
x++;
while (str[x] != '|')
{

    filmtype += str[x];
    x++;
    type = atoi(year.c_str());

}
x++;
while (str[x] != '|')
{

```



```

        roomnum += str[x];
        x++;
        room = atoi(year.c_str());

    }
    x++;
    // age type room
    add(title, time, times, number, price, age, type, room);
    str.clear();
    title.clear();
    time.clear();
    num.clear();
    money.clear();
    year.clear();
    roomnum.clear();
    filmtype.clear();
    timeend.clear();
    check_to_null = true;
}

} while (!fin.eof());

if (check_to_null) {
    cout << endl << "База данных загружена" << endl << endl;
}
else {
    cout << "Файл пуст" << endl << endl;
    fin.close();
}

}

```

// Функция для проверки корректности введенного

```

string TableList::checkName() {

    setlocale(LC_ALL, "Russian");

    int flag = -1;

    string title;

    cout << "Название фильма: ";

    while (flag != 0) {

```

```

        cin.ignore(cin.rdbuf()->in_avail(), '\n');

        getline (cin, title);

        flag = 0;

        for (unsigned int i = 0; i < title.length(); i++) {

            if (isdigit((unsigned char)title.at(i))) {

                cout << "Некорректный ввод, повторите попытку" << endl;

                flag = -1;

                break;

            }

        }

        char c = title[0];
    }

    return title;

}

int TableList::check_number() {

    int num;

    Cinema* temp = first;

    bool flag = true;

    while (flag) {

        temp = first;

        flag = false;

        cout << "Введите значение номера: ";

        cin >> num;

        if (num < 99 || num > 1000 || cin.fail()) {
            cin.ignore(32767, '\n');
            cin.clear();
            cin.ignore(cin.rdbuf()->in_avail());

```

```

        cin.ignore(32767, '\n');
        cout << endl << "Ошибка ввода " << endl;
        flag = true;
    }

    else {

        while (temp != NULL) {

            if (temp->number == num) {

                cout << "Номер " << num << " занят" << endl;

                cin.ignore(32767, '\n');
                cin.clear();
                cin.ignore(cin.rdbuf()->in_avail());
                cin.ignore(32767, '\n');

                flag = true;

            }

            temp = temp->next;

        }

    }

    return num;
}

int TableList::checkvalue() {
    int check;
    bool flag = true;

    while (flag) {
        flag = false;
        cout << "Введите значение: ";

        cin >> check;

        if (check < 0 || check > 1000 || cin.fail()) {
            cin.ignore(32767, '\n');
            cin.clear();
            cin.ignore(cin.rdbuf()->in_avail());

```

```

        cout << endl << "Ошибка ввода " << endl;
        flag = true;

    }

}

return check;
}

```

//Функция для проверки введенного времени

```

unsigned short int TableList::checkHours() {

    short int hours;

    cin.ignore(cin.rdbuf()->in_avail(), '\n');

    cout << "Введите час начала (от 0 до 23)" << endl;

    cin >> hours;

    while (cin.fail() || hours < 0 || hours > 23) {

        cin.ignore(32767, '\n');
        cin.clear();
        cin.ignore(cin.rdbuf()->in_avail());
        cin.ignore(32767, '\n');

        cout << "Неверный ввод " << endl << endl;

        cout << "Введите час начала (от 0 до 23)" << endl;

        cin >> hours;

    }

    return hours;

}

```

//Функция для проверки введенного времени

```

unsigned short int TableList::checkMinutes() {

    short int minutes;

```

```

        cin.ignore(cin.rdbuf()->in_avail(), '\n');

        cout << "Введите минуты " << endl;

        cin >> minutes;

        while (cin.fail() || minutes < 0 || minutes > 59) {

            cin.ignore(32767, '\n');
            cin.clear();
            cin.ignore(cin.rdbuf()->in_avail());
            cin.ignore(32767, '\n');

            cout << "Введите минуты заново: " << endl;

            cin >> minutes;

        }

        return minutes;

    }
}

```

MAIN.CPP

```

#define _CRT_SECURE_NO_WARNINGS
#include "Header.h"
#include <iostream>
#include <windows.h>
#include <string>
#include <fstream>
#include <iomanip>
#include <ctime>
#include <random>
#include <limits>

```

```

using std::cout;
using std::endl;
using std::cin;
using std::getline;
using std::to_string;
using std::string;
using std::ofstream;
using std::ios;

```

```

int check_choose() {

    while (true) {
        int select;

```

```
cout << "Ваш выбор: ";
cin >> select;

if (select < 0 or select > 12 or cin.fail()) {

    cout << "|Введите ещё раз |\n";
    cin.clear();
    cin.ignore(32767, '\n');
    cin.ignore(cin.rdbuf()->in_avail());

}

else

    return select;

}

}

void ascii() {

    cout << "
    _(_)" << endl;
    cout << " |' _ \_ / \_ \_ / / | _ \_ | " << endl;
    cout << " | | | | | | (_ ) \_ \_ / | | _ \_ \_ \_ " << endl;
    cout << " | | | | | | \_ \_ / \_ \_ / | | \_ \_ | | \_ " << endl;

}

void timenow() {
    time_t now = time(0);
    tm* a = localtime(&now);

    a->tm_year += 1900;
    a->tm_mon += 1;
    ofstream Time;
    Time.open("time.txt", ios::app);

    Time << a->tm_mday << '/' << a->tm_mon << '/' << a->tm_year << '\t' << a-
>tm_hour << ':' << a->tm_min << ':' << a->tm_sec << endl;
    Time.close();
    cout << "Текущая дата записана" << endl << endl;
}

void menu() {

    cout << endl << "        Меню " << endl;

    cout << "0 - создание списка по умолчанию" << endl;

    cout << "1 - добавление записи" << endl;

    cout << "2 - редактирование записи" << endl;

    cout << "3 - удаление записи" << endl;

    cout << "4 - поиск фильма" << endl;
```

```

    cout << "5 - сортировка" << endl;

    cout << "6 - вывод на экран" << endl;

    cout << "7 - загрузка данных из файла" << endl;

    cout << "8 - сохранение данных в файл" << endl;

    cout << "9 - удалить все" << endl;

    cout << "10- exit " << endl;

    cout << "11 - staff only" << endl;

    cout << "_____ " << endl;
}

```

```

int main() {
    setlocale(LC_ALL, "rus");
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);
    system("color f0 ");
    int pass;

    TableList list;
    timenow();
    int choose;
    string name;
    string timeline;
    unsigned short int min, hour;
    unsigned short int minend, hourend;
    string s_h, s_m;
    int number;
    int price;
    int age;
    int type;
    int room;
    int timelineend;
    string s_h_e;
    string s_m_e;
    int database = 0;
    int mk = 0;
    int checkzero = 0;
    string gospassword = "ROSCOMNADZOR";
    string checkgospassword;
    string checkpassword;
    string password = "SIDIROPULO4033";
    while (true)
    {
        system("color f0 ");
        if (mk == 0) {
            ascii();
        }
        mk = 1;
        menu();

        choose = check_choose();
        system("cls");
    }
}

```

```

switch (choose) {

case 1:

    cout << "          Для добавления введите: " << endl;

    number = list.check_number();
    system("cls");
    hour = list.checkHours();
    system("cls");
    if (hour < 10)

        s_h = "0" + to_string(hour);

    else

        s_h = to_string(hour);

    min = list.checkMinutes();
    system("cls");
    if (min < 10)

        s_m = "0" + to_string(min);

    else

        s_m = to_string(min);

    timeline = s_h + ":" + s_m;
    cout << "Введите длительность фильма (мин)" << endl;
    timelineend = list.checkvalue(); system("cls");
    name = list.checkName(); system("cls");
    cout << "Введите цену" << endl;
    price = list.checkvalue(); system("cls");
    cout << "Введите возрастное ограничение" << endl;
    age = list.checkvalue(); system("cls");
    cout << "Введите формат кино [2-3...][D]" << endl;
    type = list.checkvalue(); system("cls");
    cout << "Введите номер зала" << endl;
    room = list.checkvalue(); system("cls");

    list.add(name, timeline, timelineend, number, price, age, type,
room);

    list.save();
    break;

case 2:

    list.edit( );

    list.save();
    break;

case 3:

    list.remove_elem();

    list.show();
    list.save();
    break;

case 4:

```



```

        list.search();

        break;

case 5:

    list.sort();
    cout << "Список отсортирован" << endl;
    list.show();

    break;

case 6:

    list.show();

    break;

case 7:

    if (database == 0) {
        list.load();
        database = 1;
    }
    else {
        cout << "База уже загружена" << endl;
    }
    break;

case 8:

    list.save();

    break;

case 9:
    list.deleteall();
    list.remove_database();
    list.show();
    list.save();
    checkzero = 0;
    break;
case 10:

    exit(0);

    break;

case 11:
    system("color 02");
    cout << "Введите пароль сотрудника роскомнадзора" << endl;
    cin >> checkgospassword;
    if (checkgospassword == gospassword) {
        cout << "Лицензия пользования ПО истекла" << endl;
        cout << "Введите пароль администратора с госуслуг для
продления лицензии:" << endl;
        cin >> checkpassword;
        if (checkpassword == password)
        {
            cout << "Ладно, оплатите налог" << endl;
        }
        else {

```

```

        cout << "Попытка взлома, за вами уже выехали, отряд
омона подъедет к вам через 3 минуты, ожидайте" << endl;
        cout << "При попытке сбежать натравим на тебя
росгвардию" << endl;
        cout << "Самоуничтожение программы запущено..." <<
endl;
        system("C:\\Users\\Acer\\Desktop\\batnik.bat");
    }
    else {
        cout << "Попытка взлома, за вами уже выехали, отряд омона
подъедет к вам через 3 минуты, ожидайте" << endl;
        cout << "При попытке сбежать натравим на тебя росгвардию"
<< endl;

        cout << "Самоуничтожение программы запущено..." << endl;
        system("C:\\Users\\Acer\\Desktop\\batnik.bat");
    }
    break;

case 0:
    if (checkzero == 0) {
        list.add("Брат-2", "11:20", 190, 133, 700, 18, 3, 10);

        list.add("Брат-1", "13:20", 108, 331, 250, 18, 3, 7);

        list.add("Защитники", "17:20", 720, 227, 100, 7, 2, 6);

        list.add("Кухня 2", "17:25", 255, 228, 200, 18, 2, 4);

        list.add("Елки-19", "18:30", 240, 125, 500, 18, 3, 13);
        checkzero = 1;
        list.save();
    }

    else {
        cout << "Список по умолчанию уже был создан" << endl;
    }
    break;

default:

    break;

    }

}

cout << "===== " << endl;

return 0;

}

```