Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
 ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет инновационного непрерывного образования

Кафедра экономической информатики

Дисциплина: Основы конструирования программ

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовой работе

на тему

**Разработка программы учета**

**КАРТОТЕКИ ВИДЕОФИЛЬМОВ**

Студент:

гр.­­­­­ 692351 Юруш Е.В.

Руководитель:

Н.П. Мытник

Минск 2017

## Содержание

[Задание по курсовой работе 3](#_Toc499478188)

[Введение 5](#_Toc499478189)

[1. Теоретические сведения об используемых алгоритмах 6](#_Toc499478190)

[2. Описание организации структур хранимых данных 8](#_Toc499478191)

[3. Создание пользовательских функций приложения 10](#_Toc499478192)

[4. Функциональная схема программы 12](#_Toc499478193)

[5. Описание работы программы 14](#_Toc499478194)

[5.1. Модуль администратора 15](#_Toc499478195)

[5.2. Модуль пользователя 19](#_Toc499478196)

[Заключение 21](#_Toc499478197)

[Список использованных источников 22](#_Toc499478198)

[ПриложениеА Листинг кода с комментариями 23](#_Toc499478199)

[Файл main.c 23](#_Toc499478200)

[Файл header.h 28](#_Toc499478201)

[Файл list.c 29](#_Toc499478202)

## Задание по курсовой работе

Разработать программу учета картотеки видеофильмов.

Сведения о фильмах содержат: название фильма, жанр, характеристика фильма, год выпуска, актёры.

Необходимо реализовать функции добавления новых поступлений, удаление старых фильмов, редактирование записей и т.д. Выдавать по запросу информацию о фильмах по признаку (героям, жанру, году выпуска и др.).

**Требования к программной части проекта:**

1. Разработать функциональную схему задачи, а также схемы двух модулей по указанию руководителя.
2. Создать программу (консольный режим), выполнив функциональное разбиение, реализующую следующие функции работы с бинарным/текстовым файлом. Предусмотреть создание меню 1-го уровня со следующими пунктами:
3. Вход под администратором
4. Вход под пользователем
5. Выход

Учетные записи администратора и обычного пользователя должна содержать логин и пароль. Сведения об учетных записях должны храниться в отдельном файле, желательно предусмотреть механизм шифрования/дешифрования логинов/паролей.

1. Меню 2-го уровня:

* Для администратора:

1. Создание/открытие файла с данными
2. Добавление записи
3. Редактирование записи
4. Удаление записи
5. Просмотр всех данных в табличной форме
6. Различные процедуры поиска и фильтрации данных
7. Управление пользователями
8. Выход в меню 1-го уровня

* Для обычного пользователя:

1. Открытие файла с данными
2. Просмотр всех данных в табличной форме
3. Различные процедуры поиска и фильтрации данных
4. Выход в меню 1-го уровня
5. Данные должны быть организованы в структуре (struct), не менее 7-ми полей, обязательно предусмотреть использование вложенных структур.
6. Предусмотреть обработку исключительных ситуаций
7. Построение программного кода должно соответствовать правилам, определенным в документе «Code Conventions» для соответствующего языка (технологии).
8. Предусмотреть использование различных встроенных функций для работы со строками

## Введение

Выполение данной курсовой работы согласно заданию заключается в разработке консольного приложения для учёта картотеки видеофильмов. Сведения о фильмах должны содержать следующую информацию: название фильма, жанр, характеристика фильма, год выпуска, актёры. Необходимо реализовать функции добавления новых поступлений, удаление старых фильмов, редактирование, предоставления информации по запросу по следующим признакам жанру, году выпуска, наименованию.

В первой части курсовой работы приводится описание двух используемых алгоритмов - поиска и сортировки, а также их реализация. Для организации поиска использовался алгоритм последовательного ( линейного) поиска. Для организации сортировки использовался алгоритм сортировки вставками. Данные алоритмы были использованы в силу простоты их реализации и предполагаемого относительно небольшого количества элементов списка.

Во второй части описывается организация структур хранимых данных. Информация о фильмах хранится во время выполнения программы в двух структурах, состоящих из соответственно 9 и 3 элементов. Меньшая структура является вложенной. Информация о пользователях хранится непосредственно в текстовом файле в следующем виде: роль (администратор/пользователь), логин, пароль.

В третьей части приводится описание пользовательских функций. Всего реализовано 22 пользовательские функции. Четвёртая часть курсовой работы содержит функциональную схему приложения.

В пятой части курсовой работы описан ход работы приложения. Подробно описаны его команды.

В приложении к курсовой работе приведён листинг кода с комментариями.

## Теоретические сведения об используемых алгоритмах

Опишем некоторые из используемых алгоритмов в данной курсовой работе.

**Сортировка**

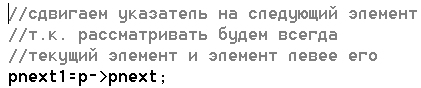
Для организации сортировки используется алгоритм сортировки вставками. Сортировка вставками – [алгоритм сортировки](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D1%81%D0%BE%D1%80%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B8), в котором элементы входной последовательности просматриваются по одному, и каждый новый поступивший элемент размещается в подходящее место среди ранее упорядоченных элементов[1]

На вход алгоритма подаётся последовательность n чисел: a1, a2, a3…,an. Сортируемые числа также называют ключами. Входная последовательность на практике представляется в виде массива с n элементами. На выходе алгоритм должен вернуть перестановку исходной последовательности a1’, a2’, a3’…,an’, чтобы выполнялось следующее соотношение a1’≤ a2’≤ a3’≤…≤an’. [2]

В начальный момент отсортированная последовательность пуста. На каждом шаге алгоритма выбирается один из элементов входных данных и помещается на нужную позицию в уже отсортированной последовательности до тех пор, пока набор входных данных не будет исчерпан. В любой момент времени в отсортированной последовательности элементы удовлетворяют требованиям к выходным данным алгоритма [2].

Кратко опишем реализацию данного алгоритма в курсовой работе. Сортировка реализована в функции sortByFeature, в которой находится головной алгоритм сортировки и в функции insInv, в которой происходит перестановка элементов массива структур местами. Рассмотрим подробней функцию sortByFeature.

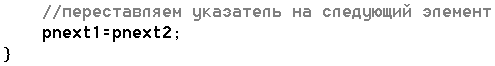
Функция на вход принимает указатель на двусвязный массив структур, который необходимо отсортировать. Первоначально, перед входом в основной цикл сортировки мы сдвигаем указатель с первого на второй элемент, т.к. в самом цикле всегда будем рассматривать теккущий элемент и элемент левее его:



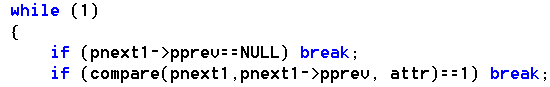
После этого мы входим в основной цикл в котором происходит последовательный просмотр элементов, до момента когда указатель на следующую структуру будет пустой:



…(тело цикла)



Далее вложенным циклом пока рассматриваемый элемент не встанет на своё место в выравненной в правильном порядке части массива, выполняем перестановку рассматриваемого и предыдущего элемента. Перестановки заканчиваются в случае если элемент стал первым в списке или если элемент левее его меньше:



…(перестановка)



Сама перестановка реализована в функции insInv, данная функция принимает указатель на предыдущий элемент и на текущий элемент и меняем указатели таким образом что расположение в памяти элементов не меняется, но меняются указатели на данные элементы и на элементы которые находятся правее и левее их.

**Поиск**

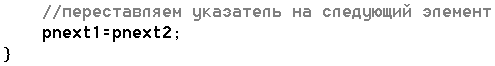
Для организации поиска используется алгоритм последовательного ( линейного) поиска. Линейный, последовательный поиск – это [алгоритм](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC) нахождения заданного значения произвольной функции на некотором отрезке. Данный алгоритм является простейшим алгоритмом поиска и, в отличие, например, от [двоичного поиска](https://ru.wikipedia.org/wiki/%D0%94%D0%B2%D0%BE%D0%B8%D1%87%D0%BD%D1%8B%D0%B9_%D0%BF%D0%BE%D0%B8%D1%81%D0%BA), не накладывает никаких ограничений на функцию и имеет простейшую реализацию. Поиск значения функции осуществляется простым сравнением очередного рассматриваемого значения (как правило, поиск происходит слева направо, то есть от меньших значений аргумента к большим) и, если значения совпадают (с той или иной точностью), то поиск считается завершённым.

Данный алоритм поиска был использован в силу простоты его реализации и предполагаемой небольшой величины элементов среди которых будет производится поиск.

Кратко опишем реализацию данного алгоритма в курсовой работе. Поиск реализован в функции find. Функция принимает на вход указатель на первый элемент связанного списка, в котором необходимо произвести поиск. В процессе выполнения функция уточняет у пользователя критерий поиска и эталонное значение, по которому необходимо произвести поиск. После этого мы входим в основной цикл в котором происходит последовательный просмотр элементов, до момента когда указатель на следующую структуру будет пустой:



…(тело цикла)



В теле цикла мы производим сравнение текущего рассматриваемого элемента с эталонным и, в случае равенства, выводим информацию о найденном элементе на экран. К примеру, если поиск производится по наименованию фильма, то условие сравнения будет выглядеть следующим образом:



Таким образом для сравнения наименований фильмов с эталоном используется библиотечная функция strcmp. Вывод на экран осуществляется с помощью специальной пользовательской функции:



## Описание организации структур хранимых данных

Для представления в программе информации о фильме вводится структура Film:

typedef struct tagFilm

{

int filmNmb;

char fname[MAXFNAME];

char genre[GENRE];

int year;

int budget;

int mark;

struct tagActor actors[5];

struct tagFilm \*pnext, \*pprev;

} Film;

Структура Film содержит следующие поля (Таблица 2.1):

Таблица 2.1 Описание полей структуры Film

|  |  |  |
| --- | --- | --- |
| **Наименование поля** | **Предназначение поля** | **Тип поля** |
| filmNmb | идентификатор фильма | int |
| fname[MAXFNAME] | название фильма | char |
| genre[GENRE] | жанр | char |
| year | год фильма | int |
| budget | бюджет фильма | int |
| mark | характеристика фильма по 10-ти бальной системе | int |
| actors[5] | актёры исполнившие главные роли | Struct tagActor |
| \*pnext | ссылка на следующую связанную структуру | Struct tagFilm |
| \*pprev | ссылка на предыдущую связанную структуру | Struct tagFilm |

Отдельно опишем некоторые поля. Тип полей «название фильма» и «жанр фильма» - строка, их длинна заранее определена как соответственно 25 и 15 символов. Поле actors[5] содержит в себе массив структур типа tagActor. В каждом элементе данного массива – структуре tagActor, хранится информация об актёре.

struct tagActor

{

char name[ACTORSNAME];

int age;

char desc[100];

};

Опишем структуру tagActor. Она содержит в себе следующие поля: name[ACTORSNAME] – поле содержащее имя, фамилию актёра, тип строка длинной 25 символов; age – поле содержащее возраст актёра, типа int; desc[100] поле содержащее описание роли и другую необходимую информацию об актёре, типа строка длинной в 100 символов.

При сохранении списка фильмов в файл происходит запись информации в текстовом виде в файл FilmLibrary.dat. Данный файл установлен по умолчанию, он создаётся при первом входе в программу и находится в папке с самой программой. Пользователь может создать/сменить файл для сохранения данных.

Информация о пользователях хранится непосредственно в текстовом файле в следующем виде: роль (администратор/пользователь), логин, пароль. Управление данной информацией (создание, удаление, редактирование пользователя) происходит способом редактирования данного файла.

## Создание пользовательских функций приложения

Для описания пользовательских функций составим таблицу:

|  |  |
| --- | --- |
| **Объявление функции** | **Описание функции** |
| **void getUser(Film \*p)** | Функция для приёма значений от пользователя. Используется при создании новой записи о фильме. Функция последовательно запрашивает данные о новом фильме и записывает их во во временную структуру. |
| **Film\* addEnd(Film \*p, Film \*end, int stnumb)** | Функция для связывания временной структуры созданной вышеобозна-ченной функцией **getUser** с имеющимся списком структур. |
| **int editStd(Film \*p, Film \*\*pend, Film \*\*pbegin)** | Функция для редактирования выбранного пользователем фильма. С помощью данной функции пользователь может редактировать/удалить/добавить актёров/редактировать актёров/удалить актёра. |
| **int delActors(Film \*p)** | Функция удаления выбранного актёра. |
| **int editActors(Film \*p)** | Функция редактирования выбранного актёр. |
| **int printActors(Film \*p)** | Функция печати актёров по выбранному фильму на экран. |
| **int addActors(Film \*p)** | Функция добавления актёров выбранному фильму. |
| **int deleteStd(Film \*p, Film \*\*pend, Film \*\*pbegin)** | Удаление выбранного фильма из списка. |
| **void saveList(Film \*p, char \*file)** | Функция записи списка фильмов в файл. |
| **void loadList(Film \*p, char \*file, Film \*\*pend, Film \*\*pbegin)** | Ф-я загрузки списка фильмов из файла. |
| **void printList(Film \*p)** | Ф-я печати списка фильмов на экран. |
| **void loadList(Film \*p, char \*file, Film \*\*pend, Film \*\*pbegin)** | Функция загрузки списка фильмов из файла. |
| **void print(Film \*p)** | Функция печати строки списка фильмов на экран. |
| **void find(Film \*p)** | Функция поиска необходимого фильма. Поиск может осуществляться по названию/по жанру/по году выпуска. |
| **Film\* insInv(Film \*p1, Film \*p2)** | Функция для смены местами элементов динамического массива. |
| **void sortByFeature(Film \*p, Film \*\*begin, Film \*\*end)** | Функция для сортировки включением. Сортировка может выполняться по ID фильма/по названию/по жанру/по году выпуска. |
| **int compare(Film \*p1, Film \*p2, int attr)** | Вспомогательная функция для сортировки элементов. Функция выполняет сравнение двух структур по заданному критерию оценки. |
| **int checkLogPas(int k)** | Функция для проверки введённого ползователем логина, пароля. |
| **void manageUsers()** | Функция для управления учётными записями пользователей: просмотр всех учётных записей пользователей/добавление новой учётной записи/удаление учётной записи. |
| **void fileManager(Film \*p, char \*file, Film \*\*pend, Film \*\*pbegin)** | Функция для управление файлом базы данных фильмов. Фыполняемый функционал: сохранить произведённые изменения/сменить файл хранения данных/создать файл для хранения данных. |
| **void check(int k, int \*n)** | Функция для приёма номера команды от пользователя я (при вооде с клавиатуры)с проверкой вводимого значения. |
| **void getLimStr(char \*c, int l, int lim)** | Функция для приёма строкового значения от пользователя (при вооде с клавиатуры) с проверкой вводимого значения. |

## Функциональная схема программы

**да**

**нет**

**выполнен первый вход?**

**да**

**создание БД и пользователя администратора**

**выход**

**выбор пользователя**

**выбор пользователя или выход**

**проверка ввода логина/пароля**

**администратор**

**пользователь**

**под какой ролью выполнен вход**

**отображение главного меню админ-тора, считывание введённого пользователем № команды**

**отображение главного меню пользователя, считывание введённого пользователем № команды**

**6**

**5**

**4**

**3**

**2**

**1**

**Сорт-ка списка фил-в по указ. признаку**

**Вывод списка актёров на экран**

**Вывод списка фильмов на экран**

**Управление файлом базы данных**

**Редактировать запись о фильме**

**Добавить фильм**

**1**

**Вывод списка фильмов на экран**

**Сорт-ка списка фил-в по указ. признаку**

**Вывод списка актёров на экран**

**2**

**3**

**4**

**Поиск фильма по указанному полю**

**5**

**Сменить файл базы данных фильмов**

**9**

**8**

**7**

**Поиск фильма по указанному полю**

**6**

**Управление учётными записями**

**Выход в меню выбора пользователя**

**Выход в меню выбора пользователя**

## Описание работы программы

При запуске программа выводит на экран приветствие и пытается найти и открыть базу данных пользователей. Если база не найдена, то программа предлагает создать её. При создании новой базы необходимо добавить в неё учётную запись администратора.

Поиск и создание файла базы данных пользователей производится в текущей директории программы. Имя файла прописано в коде программы: «FilmLibrary.dat».

Если база данных успешно открыта, то программа проверяет в ней наличие учётных записей администратора и, если их нет, предлагает добавить (рисунок 5.1)

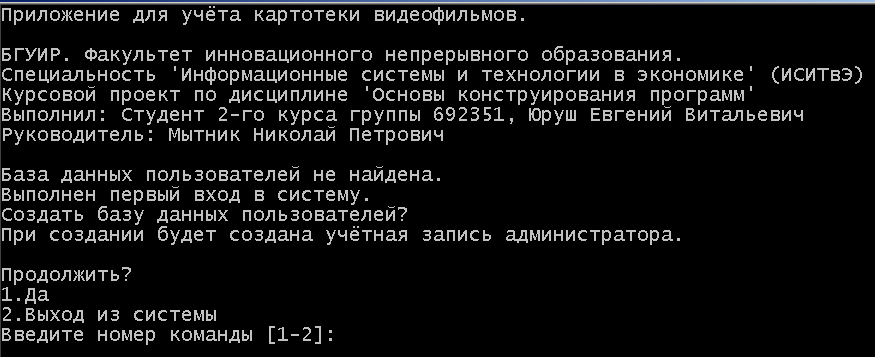


Рисунок 5.1 – Создание базы данных пользователей при запуске программы

После создания учётной записи администратора программа отображает меню выбора роли пользователя для входа. (Рисунок 5.2) На данном этапе пользователь может выбрать под какой ролью он хочет войти в программу, если это не первый вход в программу. При первом входе пользователь сможет зайти в систему только под ролью администратора. Для того чтобы зайти в программу под ролью пользователя, её необходимо завести.

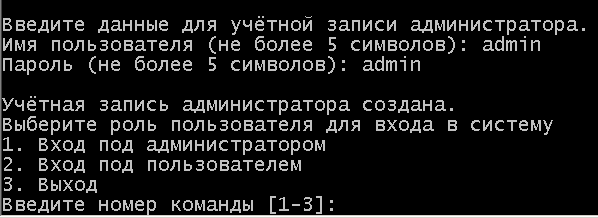


Рисунок 5.2 – Меню выбора роли пользователя для входа

После выбора роли пользователя для входа необходимо пройти проверку логина пароля. В случае неверного ввода программа оповестит пользователя об этом и предложит ввести логин пароль ещё раз или вернуться в меню выбора роли для входа.

После успешного ввода логина пароля программа отобразит меню для пользователя в зависимости от роли.

### Модуль администратора

Меню администратора выглядит следующим образом (Рисунок 5.3):

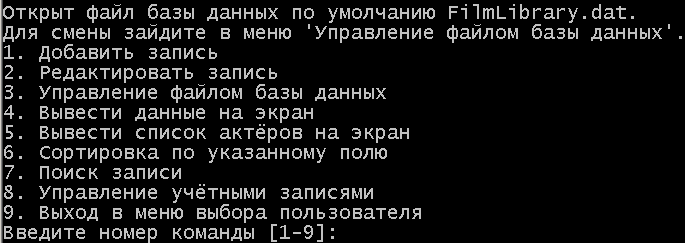


Рисунок 5.3 – Меню администратора

При входе в меню администратора программа оповещает о том что открыт файл базы данных по умолчанию и что для его смены необходимо зайти в меню «Управление файлом базы данных», это можно увидеть на вышеприведённом рисунке.

Последовательно рассмотрим команды меню администратора. При выборе команды «Добавить запись» пользователю предлагается ввести новую запись о фильме. Запрашивается информация о новом фильме (Рисунок 5.4).

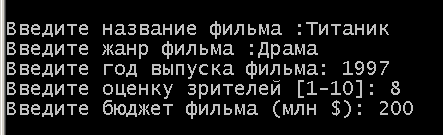


Рисунок 5.3 – Ввод нового фильма

После ввода информации о новом фильме он добавляется в список.

При выборе команды «Редактировать запись» пользователю предоставляется возможность выбора фильма, запись о котором будет редактироваться, а также необходимое действие по редактированию (Рисунок 5.4)

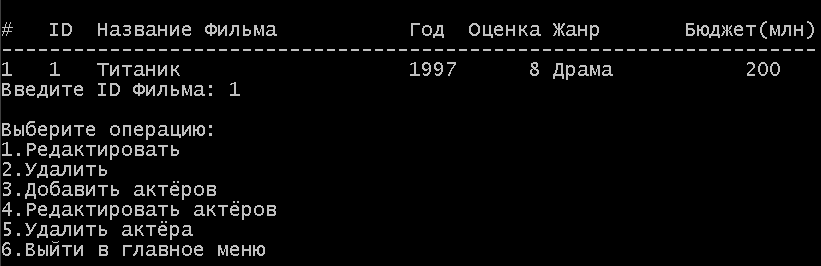


Рисунок 5.4 Редактирование фильма

При выборе операции «Редактировать» пользователю будет предложено переввести информацию по фильму для текущей записи, при выборе операции «Удалить» запись о выбранном фильме будет удалена, при выборе операции «Добавить актёров» пользователю будет прелложено ввести информацию о актёре для выбранного фильма (Рисунок 5.5). После ввода будет предложено ввести ещё одного актёра или закончить ввод актёров.

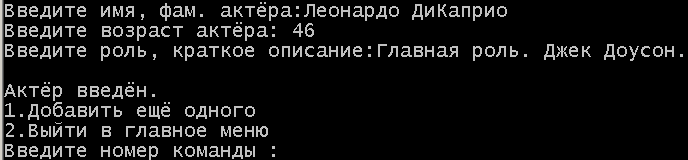


Рисунок 5.5 Ввод информации об актёре

Следующая команда в списке «Редактировать актёров», она позволяет выбрать актёра и ввести информацию по нему заного (Рисунок 5.6). Следующая команда «Удалить актёра» позволяет выбрать необходимого актёра и удалить запись о нём.

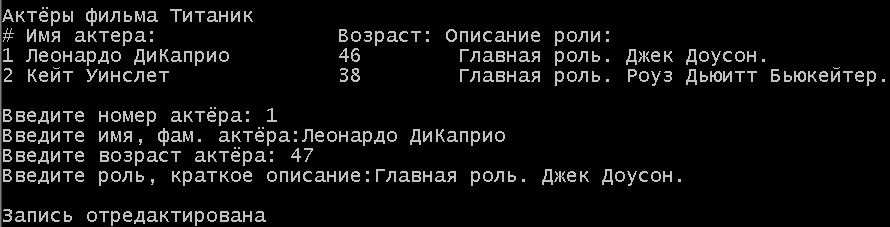


Рисунок 5.6 Редактирование записи об актёре

Рассмотрим следующую команду главного меню «Управление файлом базы данных». При выборе данной команды откроется меню для работы с файлом базы данных (Рисунок 5.7), рассмотрим данные команды.

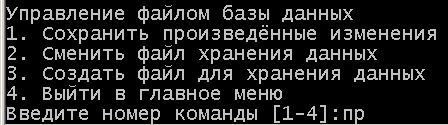


Рисунок 5.7 Меню для работы с файлом базы данных

Команда «Сохранить произведённые изменения» позволяет сохранить изменения, внесённые в список фильмов, в файл.

Команда «Сменить файл хранения данных» позволяет сменить файл хранения списка фильмов, после выбора данной команды пользователю предлагается ввести новую директорию и имя файла. В случае если программа не сможет открыть файл для работы с ним, пользователю будет отображена ошибка загрузки файла и файл не будет сменён. В случае успешного открытия файла, программа начинает с ним работать.

Команда «Создать файл для хранения данных» позволяет ввести директорию и имя нового файла для хранения данных, при этом будет создан файл и программа начнёт работу с ним.

Следующая команда главного меню, которую мы рассмотрим это «Вывести данные на экран», данная команда позволяет вывести весь список фильмов на экран (Рисунок 5.8).



Рисунок 5.8 Вывод списка фильмов на экран

Рассмотрим команду меню «Сортировка по указанному полю». Данная команда позволяет по одному из критериев (идентификатор/название/жанр/год выпуска фильма) произвести сортировку. Приведём пример выполнения сортировки по названию фильма (Рисунок 5.9):

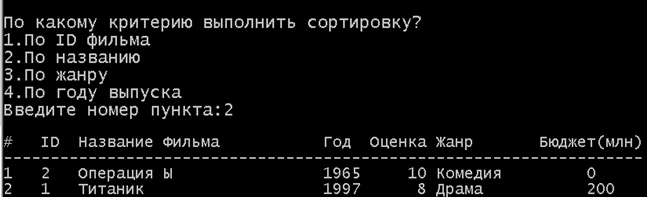


Рисунок 5.9 Сортировка фильмов по названию фильма

Следующая команда которую мы рассмотрим – «Поиск записи». Данная команда позволяет определённому критерию (названию/по жанру/году выпуска фильма) найти фильмы в списке и вывести их на экран. Приведём пример поиска фильма по жанру (Рисунок 5.10). После выбора критерия поиска программа запрашивает эталон по которому необходимо выполнить поиск. после ввода эталона найденные записи о фильмах будут выведены на экран.

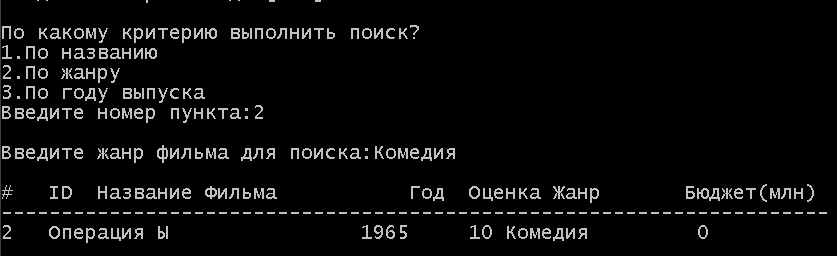


Рисунок 5.10 Поиск фильма по жанру

Рассмотрим команду «Управление учётными записями». После выбора данной команды пользователю отобразится меню (Рисунок 5.11). Рассмотрим команды данного меню. Команда «Просмотр всех учётных записей» позволяет просмотреть

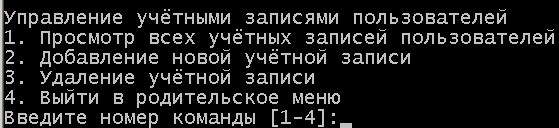


Рисунок 5.11

все учётные записи заведённые в системе. Приведём пример выполнения данной команды (Рисунок 5.12):



Рисунок 5.12 Вывод на экран всех пользователей

Команда «Добавление новой учётной записи» позволяет добавить новую учётную запись пользователя. При её выборе пользователю необходимо выбрать добавляемую роль (пользователь/администратор) и ввести логин/пароль для нового пользователя (Рисунок 5.13)

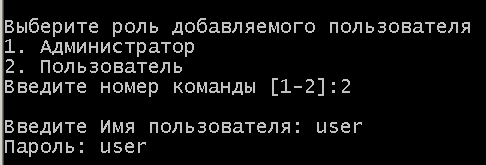


Рисунок 5.13 Ввод нового пользователя

Команда «Удаление учётной записи» позволяет выбрать пользователя из списка и удалить.

Последняя команда главного меню «Выход в меню выбора пользователей». Она позволяет выйти из главного меню в меню выбора пользователя (Рисунок 5.14) из которого можно выйти из программы, используя команду «Выход».

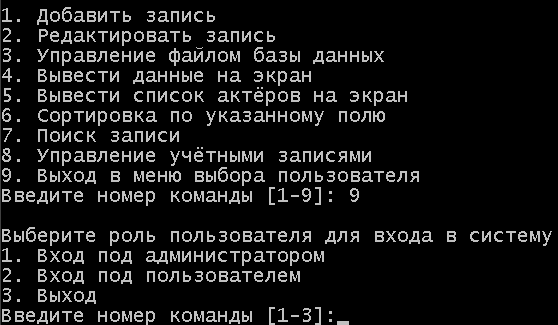


Рисунок 5.14 Выход в меню выбора пользователей

### Модуль пользователя

Меню пользователя выглядит следующим образом (Рисунок 5.15):

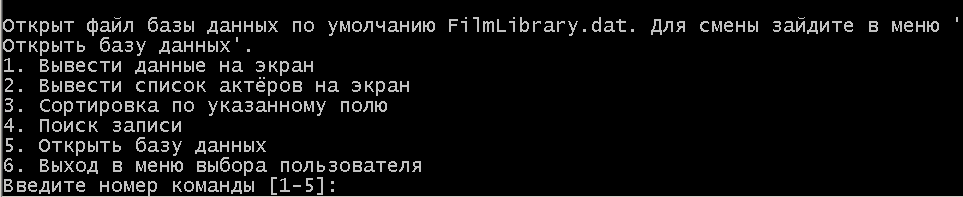


Рисунок 5.15 Меню пользователя

При входе в меню пользователя программа оповещает о том что открыт файл базы данных по умолчанию и что для его смены необходимо зайти в меню «Управление файлом базы данных», это можно увидеть на вышеприведённом рисунке.

Меню пользователя отличается от меню администратора тем что оно ограничено в своих действиях:

1. информацию о фильмах и актёрах можно только просматривать, редактировать нельзя
2. нельзя редактировать пользователей
3. нельзя управлять файлом базы данных, можно только открыть новый для просмотра

Команды меню пользователя уже рассмотрены ранее в модуле администратора, они ничем не отличаются, поэтому рассматривать их повторно не будем.

## Заключение

В данной курсовой работе согласно заданию разработано консольное приложение для учёта картотеки видеофильмов. Приложение разработано на языке Си в IDE-среде Microsoft Visual Studio 2005.В приложении реализованы следующие функции:

- добавления/удаление/редактирование записей о фильмах;

- записи о фильмах содержат следующую информацию: идентификатор фильма, название, жанр, год выпуска, бюджет, оценка зрителей;

- добавление/удаление/редактирование информации об актёрах;

- записи об актёрах содержат следующую информацию: ФИО актёра, возраст, краткие сведения о роли;

- поиск записей о фильмах по различным критериям (жанру, названию, году выпуска);

- сортировка записей о фильмах по идентификатору/названию/жанру/году выпуска;

- сохранения введённой информации в текстовый файл/загрузка информации из файла;

- создание/удаления/смена текстового файла для хранения информации о фильмах;

- аутентификаци пользователя при входе в систему;

- добавление/удаление учётных записей пользователей;

- разделение доступа к функциям приложения для учётных записей пользователя и администратора.

В данной пояснительной записке описаны используемые алгоритмы, краткие теоретические сведения о них, описана организация структур хранимых данных, описаны пользовательские функции приложения, приведена функциональная схема работы приложения, а также подробно описан ход работы приложения.

## Список использованных источников

1. [Кнут Д. Э.](https://ru.wikipedia.org/wiki/%D0%9A%D0%BD%D1%83%D1%82,_%D0%94%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%B4_%D0%AD%D1%80%D0%B2%D0%B8%D0%BD) 5.2 Внутренняя сортировка// [Искусство программирования. Том 3. Сортировка и поиск](https://books.google.ru/books?id=92rW-nktlbgC&printsec=frontcover&dq=editions:spjjKVwoQ3QC&hl=ru&sa=X&ved=0ahUKEwiUjLbc88rQAhVCfiwKHUJlBQoQuwUIIjAB#v=onepage&q&f=false)  / под ред. В. Т. Тертышного (гл. 5) и И. В. Красикова (гл. 6). — 2-е изд. — Москва: Вильямс, 2007. — Т. 3. — 832 с.
2. Кормен, Т. Алгоритмы: учебник / Т. Кормен [и др.].: Москва - Санкт-Петербург - Киев 2005. - 1290 с
3. Википедия – свободная энциклопедия. [Электронный ресурс]. – 2017. – Режим доступа: <https://ru.wikipedia.org>
4. БГУИР – официальный сайт высшего учебного заведение БГУИР. [Электронный ресурс] – 2017. – Режим доступа: <http://www.bsuir.by>
5. Основы программирования на языках Си и C++ для начинающих. [Электронный ресурс] – 2017. – Режим доступа: <http://cppstudio.com/>
6. Информационный сайт о высоких технологиях [Электронный ресурс] – 2017. – Режим доступа: <http://all-ht.ru/>
7. Введение в язык си [Электронный ресурс] – 2017. – Режим доступа: <http://dfe.petrsu.ru>

## ПриложениеА Листинг кода с комментариями

## Файл main.c

//подключаем необходимые заголовочные файлы стандартной библиотеки

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include <windows.h>

//подключаем созданный заголовочный файл

#include "header.h"

//пишем главную функцию программы

void main()

{

//объявляем переменные

//переменная для выбранного пользователем номера команды

int n=0, flag=0, flag2=0, k;

char strtemp[4], login[15], pass[15];

char dir[100]={"FilmLibrary.dat"};

//перем. для временного хранения стр-ры, для передачи её из одной функции в другую

Film std;

//указатели на первый и последний элемент связанных структур

Film \*begin=NULL, \*end=NULL;

//задаём русскую локализацию

//объявляем указатель типа файл и пытаемся сразу открыть базу данных пользователей

FILE \*pf=fopen("usr.dat","r");

SetConsoleOutputCP(1251);

SetConsoleCP(1251);

printf("Приложение для учёта картотеки видеофильмов.\

\n\

\nБГУИР. Факультет инновационного непрерывного образования.\

\nСпециальность 'Информационные системы и технологии в экономике' (ИСИТвЭ)\

\nКурсовой проект по дисциплине 'Основы конструирования программ'\

\nВыполнил: Студент 2-го курса группы 692351, Юруш Евгений Витальевич\

\nРуководитель: Мытник Николай Петрович");

//1.если бд пользователей открыта, проверяем заведены ли пользователи в базе

n=0;

if (pf!=NULL) {

while(!feof(pf)) {

fgets(strtemp, 4, pf);

if (strcmp(strtemp,"adm")==0 || strcmp(strtemp,"usr")==0) n++;

}

if (n!=0) {

printf("\n");

printf("%s","\nБаза данных пользователей загружена.");

}

}

if (pf==NULL || n==0) {

remove("usr.dat");

printf("\n");

printf("%s","\nБаза данных пользователей не найдена. \nВыполнен первый вход в систему.\

\nСоздать базу данных пользователей? \nПри создании будет создана учётная запись администратора.\

\nПродолжить?\n1.Да\n2.Выход из системы\nВведите номер команды [1-2]:");

scanf("%d", &n);

if (n==1){

//создаём базу данных, указатель на неё помещаем в pf

pf=fopen("usr.dat","w");

//если файл успешно открыт/создан

if (pf!=NULL){

fputs("adm", pf);

fprintf(pf,"\n");

printf("\nВведите данные для учётной записи администратора.\nИмя пользователя: ");

strcpy(login,"");

fflush(stdin);

fgets(login,15,stdin);

login[strlen(login)-1]='\0';

fputs(login, pf);

fprintf(pf,"\n");

printf("Пароль: ");

strcpy(pass,"");

fflush(stdin);

fgets(pass,15,stdin);

pass[strlen(pass)-1]='\0';

fputs(pass, pf);

fprintf(pf,"\n");

fclose(pf);

printf("\nУчётная запись администратора создана.");

} else {

printf("Ошибка создания базы данных.");

}

} else {

printf("\nВыполнен выход из приложения.\n");

exit(0);

}

}

flag=1;

while (flag==1) {

n=0;

flag=0;

while (flag==0)

{

printf("\nВыберите роль пользователя для входа в систему\

\n1. Вход под администратором\

\n2. Вход под пользователем\

\n3. Выход\

\nВведите номер команды [1-3]:");

scanf("%d", &n);

k=0;

switch(n)

{

case 3: exit(0);

case 2: {

k=2;

break;

}

case 1: k=1;

}

flag2=0;

while (flag2==0) {

if (checkLogPas(k)==0) {

printf("Вход в систему выполнен успешно\n");

flag=1;

flag2=1;

} else {

printf("Не найдено пользователя с таким логином, паролем.\n");

printf("\nПовторить ввод?\

\n1. Да\

\n2. Сменить выбранную роль\

\n3. Выйти из программы\

\nВведите номер команды [1-3]:");

scanf("%d", &n);

switch(n)

{

case 3: exit(0);

case 2: flag2=1; break;

case 1: break;

}

}

}

}

fclose(pf);

flag=0;

if (k==1) {

loadList(begin,dir,&end, &begin);

printf("Открыт файл базы данных по умолчанию FilmLibrary.dat. Для смены зайдите в меню 'Управление файлом базы данных'.");

while (flag==0) {

printf("\n1. Добавить запись \n2. Редактировать запись \n3. Управление файлом базы данных \n4. Вывести данные на экран \n5. Вывести список актёров на экран\

\n6. Сортировка по указанному полю \n7. Поиск записи \n8. Cоздание файла с данными\

\n9. Открытие файла с данными \n10. Управление учётными записями \n11. Выход в меню выбора пользователя");

printf("\nВведите номер команды [1-11]: ");

//сохраняем введённое пользователем значение в переменной n

scanf("%d", &n);

//в зав-ти от введённого пользователем значения (1-7) запуск. соотв. функции,

//при вводе '9' (9-й команды) выходим без ошибки

switch(n)

{

case 1: //принимаем значения от пользователя во временную структуру std

getUser(&std);

//связываем заполненную польз-ем стр-ру std с имеющимися структурами

end=addEnd(&std,end,0);

//при первом заполнении стр-ры указатели на 1 и последн. структуру равны

if (begin==NULL) begin=end;

break;

case 2: //редактируем выбранную пользователем строку

editStd(begin, &end, &begin);

break;

case 3: {

fileManager(begin,&dir,&end, &begin);

printf("%s",dir);

break;

}

case 4: //печатаем список на экран

printList (begin); break;

case 5: //печатаем список актёров

printActors (begin); break;

case 6: //сортируем двусвязный список вставками

sortByFeature(begin, &begin, &end); break;

case 7: //находим необходимую строку

find(begin); break;

case 8: //управление пользователями

manageUsers(); break;

case 9: //завершаем программу без ошибки

{

flag=1;

break;

}

}

//при вводе пользователем значения не равного 1,2,3,4,5,6,7

//просим пользователя ввести команду ещё раз

}

} else if (k==2)

{

loadList(begin,dir,&end, &begin);

printf("\nОткрыт файл базы данных по умолчанию FilmLibrary.dat. Для смены зайдите в меню 'Открыть базу данных'.");

while (flag==0) {

printf("\n1. Вывести данные на экран \n2. Вывести список актёров на экран\

\n3. Сортировка по указанному полю \n4. Поиск записи \n5. Открыть базу данных \n6. Выход в меню выбора пользователя");

printf("\nВведите номер команды [1-5]: ");

printf("\n%s",dir);

//сохраняем введённое пользователем значение в переменной n

scanf("%d", &n);

//в зав-ти от введённого пользователем значения (1-7) запуск. соотв. функции,

//при вводе '9' (9-й команды) выходим без ошибки

switch(n)

{

case 1: //печатаем список на экран

printList (begin); break;

case 2: //печатаем список актёров

printActors (begin); break;

case 3: //сортируем двусвязный список вставками

sortByFeature(begin, &begin, &end); break;

case 4: //находим необходимую строку

find(begin); break;

case 5: {

FILE \*pf;

k=0;

printf("Хотите сменить базу данных?\

\n1. Да\

\n2. Нет\

\nВведите номер команды [1-2]:");

scanf("%d", &k);

switch(k)

{

case 2: loadList(begin,dir,&end, &begin); break;

case 1: {

char newdir[100]={""};

printf("Введите директорию:\n");

fflush(stdin);

fgets(newdir,100,stdin);

newdir[strlen(newdir)-1]='\0';

pf=fopen(newdir,"r");

if (pf!=NULL) {

fclose(pf);

strcpy(dir,newdir);

printf("База данных загружена.");

loadList(begin,dir,&end, &begin);

} else printf("Ошибка загрузки файла базы данных");

fflush(stdin);

break;

}

}

break;

}

case 6: //завершаем программу без ошибки

{

(flag=1);

break;

}

}

}

}

}

return 0;

}

## Файл header.h

//объявляем константу в которой храним количество символов

#define MAXFNAME 25

#define ACTORSNAME 25

#define GENRE 10

//объявляем структуру для хранения инф. об актёре

struct tagActor

{

char name[ACTORSNAME]; //имя фам актёра

int age; //возраст актёра

char desc[100]; //описание (роль и проч)

};

//объявляем структуру для хранения инф. о фильме

typedef struct tagFilm

{

int filmNmb; //ID фильма

char fname[MAXFNAME]; //название фильма

char genre[GENRE]; //жанр

int year; //год фильма

int budget; //бюджет фильма

int mark; //характеристика фильма по 10ти бальной системе

struct tagActor actors[5]; //актёры исполнившие главные роли

//в этих переменных храним ссылку на следующую, предыдущую связанную структуру

struct tagFilm \*pnext, \*pprev;

} Film;

//объявляем прототипы функций используемых в программе

void getUser(Film \*p);

Film\* addEnd(Film \*p, Film \*end, int stnumb);

int editStd(Film \*p, Film \*\*pend, Film \*\*pbegin);

int delActors(Film \*p);

int editActors(Film \*p);

int printActors(Film \*p);

int addActors(Film \*p);

int deleteStd(Film \*p, Film \*\*pend, Film \*\*pbegin);

void saveList(Film \*p, char \*file);

void loadList(Film \*p, char \*file, Film \*\*pend, Film \*\*pbegin);

void printList(Film \*p);

void print(Film \*p);

void find(Film \*p);

Film\* insInv(Film \*p1, Film \*p2);

void sortByFeature(Film \*p, Film \*\*begin, Film \*\*end);

int compare(Film \*p1, Film \*p2, int attr);

int checkLogPas(int k);

void manageUsers();

void fileManager(Film \*p, char \*file, Film \*\*pend, Film \*\*pbegin);

## Файл list.c

//подключаем необходимые заголовочные файлы стандартной библиотеки

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

//подключаем созданный заголовочный файл

#include "header.h"

//функция для приёма значений от пользователя

//вход параметр: указатель на структуру

void getUser(Film \*p)

{

Film tmp;

int i;

//поочерёдно запрашиваем и принимаем информацию от пользователя

printf("\nВведите название фильма :");

//fflush(stdin);

//fgets(tmp.fname,MAXFNAME,stdin);

//tmp.fname[strlen(tmp.fname)-1]='\0';

getLimStr(&tmp.fname, 100, MAXFNAME+1);

//printf("%s",tmp.fname);

printf("Введите жанр фильма :");

//fflush(stdin);

//fgets(tmp.genre,GENRE,stdin);

//удаляем символ завершения строки кот. добавился функцией fgets

//tmp.genre[strlen(tmp.genre)-1]='\0';

getLimStr(&tmp.genre, 100, GENRE+1);

fflush(stdin);

printf("Введите год выпуска фильма: ");

scanf("%d", &tmp.year);

fflush(stdin);

printf("Введите оценку зрителей [1-10]: ");

scanf("%d", &tmp.mark);

fflush(stdin);

printf("Введите бюджет фильма (млн $): ");

scanf("%d", &tmp.budget);

//инициализируем имена актёров пустыми значениями

for(i=0;i<5;i++) {

strcpy(tmp.actors[i].name,"");

}

//указатели на пред. след. структуру укажем позже

tmp.pnext = tmp.pprev = NULL;

//копируем структуру tmp в структуру по указателю p (вход. пар-р)

\*p = tmp;

}

//функция (Ф) для связывания заполненной польз-ем структуры с имеющимся списком

//структур. Ф принимает: заполненную структуру, указатель на последнюю

//структуру в списке, номер в списке который необходимо заполнить для

//добавляемого фильма (т.к. данная ф-я еще используется при загрузке

//списка из файла). Ф возвращает: указатель на структуру который записывается в

//переменную end - указатель на последнюю структуру

Film\* addEnd(Film \*p, Film \*end, int stnumb)

{

//выделяем память для структуры, помещаем указатель на неё в pAdd

Film \*pAdd=(Film\*)malloc(sizeof(Film));

//заполняем память по адресу pAdd переданной, через параметр указатель p, структурой

\*pAdd=\*p;

if (end==NULL) //если добавляем первый элемент

{

//в указатель end помещаем указатель на заполненную структуру

end=pAdd;

//если переданный параметр stnumb равен нулю, тогда номеру в списке присваиваем 1

//иначе значение равное переданному параметру stnumb

if (stnumb==0) pAdd->filmNmb=1;

else pAdd->filmNmb=stnumb;

}

else //если добавляем не первый элемент

{

end->pnext = pAdd;//связываем последний элемент с добавляемым

pAdd->pprev=end; //связываем добавляемый элемент с последним

//если переданный параметр stnumb равен нулю, тогда номер в списке присваиваем

//номер последней строки в списке плюс один

if (stnumb==0) pAdd->filmNmb=end->filmNmb+1;

else pAdd->filmNmb=stnumb;

end= pAdd;

}

//возвращаем указатель на последний элемент

return end;

}

//функция для редактирования выбранных пользователем строк

//вход. пар-ры: указатель на первую структуру,

//указатель на указатель последней в списке структуры,

//указатель на указатель первой в списке структуры,

int editStd(Film \*p, Film \*\*pend, Film \*\*pbegin)

{ //объявляем переменные

int n, k;

Film \*s, std;

//выводим список на экран и предлагаем выбрать номер строки

printList (p);

printf("Введите ID Фильма: ");

scanf("%d", &n);

//находим структуру номер в списке кот. равен выбранному

while(p!=NULL) {

if (p->filmNmb==n){ s=p; break;}

p=p->pnext; }

//предлагаем выбрать номер команды "что сделать со строкой"

L: printf("\nВыберите операцию: \n1.Редактировать \n2.Удалить\

\n3.Добавить актёров \n4.Редактировать актёров\

\n5.Удалить актёра \n6.Выйти в главное меню \n");

check(6, &k);

switch(k)

{

case 1: //при редактировании строки запрашиваем переввести данные

getUser(&std);//и сохраняем их в структуру std

//в выбранную пользователем строку вносим изменения

strncpy(s->fname,std.fname,sizeof(std.fname));

s->mark=std.mark; s->year=std.year;

strncpy(s->genre,std.genre,sizeof(std.genre));

s->budget=std.budget;

goto M; //выход из цикла

case 4: //запускаем функцию редактирования актёров

editActors(s); goto M;

case 3: //запускаем функцию добавления актёров

addActors(s); goto M;

case 5: //запускаем функцию добавления актёров

delActors(s); goto M;

case 2: //удаляем строку функцией deleteStd, передаём в неё

//указатель на удаляемую стр-ру, на первый и последний эл-ты списка

deleteStd(s, pend, pbegin); goto M; //и выходим из цикла

case 6: goto M; //выход из цикла в главное меню

}

goto L; //переспрашиваем номер команды если введённой значение не 1,2,3

M: return 0;//выходим без кода об ошибке

}

//функция удаления актёров

int delActors(Film \*p)

{

int i,n,k;

n=-1;

for(i=0;(i<5 && n==-1);i++){

if (strcmp(p->actors[i].name,"")!=0)

n++;

}

if (n==-1) printf("\nАктёры не найдены\n");

else {

printf("\nАктёры фильма %s \n",p->fname);

printf("# Имя актера: Возраст: Описание роли:\n");

for(k=0;k<5;k++)

{

if (strcmp(p->actors[k].name,"")!=0)

printf("%d %-25s %-9d %s \n",k+1,p->actors[k].name,p->actors[k].age,p->actors[k].desc);

}

printf("\nВведите номер актёра: ");

scanf("%d", &n);

n--;

if (n>=0 && n<5) {

strcpy(p->actors[n].name,"");

p->actors[n].age=0;

strcpy(p->actors[n].desc,"");

printf("\nАктёр удалён.");

}

else {

printf("\nНе найдено актёра с таким номером");

}

}

return 0;//выходим без кода об ошибке

}

//функция редактирования актёров фильма

//параметр указатель на структуру типа Film

int editActors(Film \*p)

{

int i,n,k;

n=-1;

for(i=0;(i<5 && n==-1);i++){

if (strcmp(p->actors[i].name,"")!=0)

n++;

}

if (n==-1) printf("\nАктёры не найдены\n");

else {

printf("\nАктёры фильма %s \n",p->fname);

printf("# Имя актера: Возраст: Описание роли:\n");

for(k=0;k<5;k++)

{

if (strcmp(p->actors[k].name,"")!=0)

printf("%d %-25s %-9d %s \n",k+1,p->actors[k].name,p->actors[k].age,p->actors[k].desc);

}

printf("\nВведите номер актёра: ");

scanf("%d", &n);

n--;

if (n>=0 && n<5) {

printf("Введите имя, фам. актёра:");

getLimStr(&p->actors[n].name, 100, ACTORSNAME+1);

fflush(stdin);

printf("Введите возраст актёра: ");

scanf("%d", &p->actors[n].age);

printf("Введите роль, краткое описание:");

getLimStr(&p->actors[n].desc, 100, 50+1);

printf("\nЗапись отредактирована");

}

else {

printf("\nНе найдено актёра с таким номером");

}

}

return 0;//выходим без кода об ошибке

}

//функция печати актёров

//параметр указатель на структуру типа Film

int printActors(Film \*p)

{

Film \*s;

int i, k, n;

//печатаем список студентов

printList (p);

//спрашиваем по какому студенту распечатать оценки

printf("Введите № фильма: ");

scanf("%d", &n);

//находим структуру номер в списке кот. равен выбранному

i=1;

while(p!=NULL) {

if (i==n){

s=p;

}

i++;

p=p->pnext;

}

n=-1;

for(i=0;(i<5 && n==-1);i++){

if (strcmp(s->actors[i].name,"")!=0)

n++;

}

if (n==-1) printf("\nАктёры не найдены\n");

else {

printf("\nАктёры фильма %s \n",s->fname);

printf("Имя актера: Возраст: Описание роли:\n");

for(k=0;k<5;k++)

{

if (strcmp(s->actors[k].name,"")!=0)

printf("%-25s %-9d %s \n",s->actors[k].name,s->actors[k].age,s->actors[k].desc);

}

}

return 0;//выходим без кода об ошибке

}

//функция добавления актёров фильму

//параметр указатель на структуру типа Film

int addActors(Film \*p)

{

int i,n,k;

//найдём свободную запись для актёра

L: n=-1;

for(i=0;(i<5 && n==-1);i++){

if (strcmp(p->actors[i].name,"")==0)

n=i;

}

if (n>-1) {

printf("Введите имя, фам. актёра:");

getLimStr(&p->actors[n].name, 100, ACTORSNAME+1);

fflush(stdin);

printf("Введите возраст актёра: ");

scanf("%d", &p->actors[n].age);

printf("Введите роль, краткое описание:");

getLimStr(&p->actors[n].desc, 100, 50+1);

//предоставляем возможность продолжить вводить оценки

printf("\nАктёр введён. \n1.Добавить ещё одного \n2.Выйти в главное меню");

printf("\nВведите номер команды :");

check(2, &k);

//выходим либо продолжаем вводить оценки

switch(k){

case 1: goto L;

case 2: goto M;

}

}

else {

printf("Нет свободных записей для добавления актёра. Измените существующего.");

}

M: return 0;//выходим без кода об ошибке

}

//удаление строки из списка, параметры идентичные функции editStd

int deleteStd(Film \*p, Film \*\*pend, Film \*\*pbegin)

{

Film \*pr=NULL, \*pn=NULL;//указатели для пред. след. эл. списка

if (p->pnext==NULL) //если удаляемая строка последняя

{

pr=p->pprev; //сохраняем в pr указатель на предыд. эл. списка

if(pr!=NULL) //если предыд. эл. есть

{

pr->pnext=NULL; //обнуляем указатель в пред. эл-те

\*pend=pr; //предыдущий элемент объявляем последним

} else //если предыд. элемента нет (т.е. элемент в списке один)

{

\*pend=NULL; //последний и первый элементы списка обнуляем

\*pbegin=NULL;

}

}

else if (p->pprev==NULL) //если удаляемая строка первая

{

pn=p->pnext; //сохраняем в pn указатель на след. эл. списка

\*pbegin=pn; //указатель на первый элемент меняем на pn

pn->pprev=NULL; //в след эл-те обнуляем указатель на пред. эл-т

}

else //если удаляемая стр. ни первая ни последняя

{

pn=p->pnext; //сохраняем в pn указатель на след. эл. списка

pr=p->pprev; //сохраняем в pr указатель на предыд. эл. списка

//связываем указ-ли предыдущ. и след. эл-та так как тек. эл-т удаляем

pn->pprev=p->pprev;

pr->pnext=p->pnext;

}

free(p); //освобождаем память по указателю тек. элемента

p=NULL; //обнуляем указатель

return 0; //выходим без кода ошибки

}

//функция записи списка в файл

//параметр: указатель на первый элемент списка и название файла

void saveList(Film \*p, char \*file)

{

int i,n,k;

//объявляем указатель pf типа FILE, открываем/создаём файл

//функцией fopen (режим символьн. записи), и помещаем указатель на файл в pf

FILE \*pf=fopen(file,"w");

//если файл успешно открыт/создан

if (pf!=NULL)

{

while(p!=NULL)//проходимся по всем элементам списка

{

//и записываем элементы структур в файл

fprintf(pf,"%d %d %d %d\n", p->filmNmb, p->mark, p->year, p->budget);

fputs(p->fname, pf);

fprintf(pf,"\n");

fputs(p->genre, pf);

fprintf(pf,"\n");

//---------------------

n=0;

for(i=0;i<5;i++){

if (strcmp(p->actors[i].name,"")!=0)

n++;

}

fprintf(pf,"%d\n", n);

if (n>0) {

for(k=0;k<5;k++) {

if (strcmp(p->actors[k].name,"")!=0) {

fprintf(pf,"%d\n", p->actors[k].age);

fputs(p->actors[k].name, pf);

fprintf(pf,"\n");

fputs(p->actors[k].desc, pf);

fprintf(pf,"\n");

}

}

}

//------------------------

p=p->pnext;

}

fclose(pf);//закрываем файл

}

}

//функция загрузки списка из файла, параметры:

//указатель на структуру типа Film, название файла, указатель на

//указ-ль на первый эл-т списка, указ-ль на указ-ль на последний эл-т списка

void loadList(Film \*p, char \*file, Film \*\*pend, Film \*\*pbegin)

{

int i, k;

Film tmp, \*ptmp;//

//объявляем указатель pf типа FILE, открываем/создаём файл

//функцией fopen (режим символьн. чтения), и помещаем указатель на файл в pf

FILE \*pf=fopen(file,"r");

//если файл успешно открыт/создан

if (pf!=NULL)

{

//очищаем текущий список структур

while(p!=NULL)

{

ptmp=p->pnext;

deleteStd(p, \*pend, \*pbegin);

p=ptmp;

}

\*pend=NULL;

\*pbegin=NULL;

//пока не достигнут конец файла читаем данные во временную структуру tmp

while(!feof(pf))

{

fscanf(pf,"%d %d %d %d\n", &tmp.filmNmb, &tmp.mark, &tmp.year, &tmp.budget);

fscanf(pf,"%25[^\r\n]\n", &tmp.fname);

fscanf(pf,"%10[^\r\n]\n", &tmp.genre);

tmp.pnext = tmp.pprev = NULL;

fscanf(pf,"%d\n", &i);

if (i>0) {

for(k=0;k<i;k++) {

fscanf(pf,"%d\n", &tmp.actors[k].age);

fscanf(pf,"%25[^\r\n]\n", &tmp.actors[k].name);

fscanf(pf,"%100[^\r\n]\n", &tmp.actors[k].desc);

}

for(k=i;k<5;k++){

strcpy(tmp.actors[k].name,"");

}

}

//запускаем функцию addEnd, передаём в неё указатель на заполненную временную структуру,

//указатель на указатель последнего элемента списка, и номер записываемого элемента в списке

//функция добавит полученную из файла структуру в список

\*pend=addEnd(&tmp,\*pend,tmp.filmNmb);

//если элемент первый то последний элемент равен первому элементу

if (\*pbegin==NULL) \*pbegin=\*pend;

}

fclose(pf);//закрываем файл

}

}

//функция вывода списка на экран принимает указатель на структуру

void printList(Film \*p)

{

int n=1;

//выводим шапку таблицы

printf("\n%s %s %s %s %s %s %s \n","# ","ID ","Название Фильма ","Год ","Оценка","Жанр ","Бюджет(млн)");

printf("%s \n","--------------------------------------------------------------------");

//циклом проходимся по элементам списка и выводим их на экран

//ниже описанной функцией print

while(p!=NULL)

{

printf("%-3d ",n);

print(p);

p=p->pnext;

n++;

}

}

//функция печати строки списка н экран

void print(Film \*p)

{

printf("%-3d %-25s %-4d %6d %-15s %-12d \n", p->filmNmb, p->fname, p->year, p->mark, p->genre, p->budget);

}

//функция поиска необходимых строк на экран, принимает указатель на первый элемент связанного списка

void find(Film \*p)

{

int k;

printf("\nПо какому критерию выполнить поиск?\

\n1.По названию\

\n2.По жанру\

\n3.По году выпуска\

\nВведите номер пункта:");

check(3, &k);

switch(k)

{

case 1: {

//вводим переменную флаг(0,1) и переменную для хранения ввода пользователя

int flag=0;

char tempname[MAXFNAME];

//запрашиваем имя фильма для поиска

printf("\nВведите наименование фильма для поиска:");

getLimStr(&tempname, 100, MAXFNAME+1);

//scanf("%s", tempname);

//циклом проходимся по элементам списка и выводим на экран строку

//с информацией по необходимому фильму

while(p!=NULL)

{

if (strcmp(p->fname,tempname)==0)

{

if (flag==0)

{

//выводим шапку таблицы

printf("\n%s %s %s %s %s %s %s \n","ID ","Название Фильма ","Год ","Оценка","Жанр ","Бюджет(млн)", "Актёры");

printf("%s \n","---------------------------------------------------------------------");

flag=1;

}

print(p);

}

p=p->pnext;

}

//не нашли фильм, сообщаем об этом пользователю

if (flag==0) printf("%s %s\n","Не найдено фильма с таким наименованием - ",tempname);

break;

}

case 2: {

//вводим переменную флаг(0,1) и переменную для хранения ввода пользователя

int flag=0;

char tempname[GENRE];

//запрашиваем имя фильма для поиска

printf("\nВведите жанр фильма для поиска:");

getLimStr(tempname, 100, GENRE+1);

//циклом проходимся по элементам списка и выводим на экран строку

//с информацией по необходимому фильму

while(p!=NULL)

{

if (strcmp(p->genre,tempname)==0)

{

if (flag==0)

{

//выводим шапку таблицы

printf("\n%s %s %s %s %s %s %s \n","# ","ID ","Название Фильма ","Год ","Оценка","Жанр ","Бюджет(млн)", "Актёры");

printf("%s \n","---------------------------------------------------------------------");

flag=1;

}

print(p);

}

p=p->pnext;

}

//не нашли фильм, сообщаем об этом пользователю

if (flag==0) printf("%s %s\n","Не найдено фильма с таким жанром - ",tempname);

break;

}

case 3: {

//вводим переменную флаг(0,1) и переменную для хранения ввода пользователя

int flag=0, year;

//запрашиваем имя фильма для поиска

printf("\nВведите год выпуска фильма для поиска:");

scanf("%d", &year);

//циклом проходимся по элементам списка и выводим на экран строку

//с информацией по необходимому фильму

while(p!=NULL)

{

if (p->year==year)

{

if (flag==0)

{

//выводим шапку таблицы

printf("\n%s %s %s %s %s %s %s \n","# ","ID ","Название Фильма ","Год ","Оценка","Жанр ","Бюджет(млн)", "Актёры");

printf("%s \n","---------------------------------------------------------------------");

flag=1;

}

print(p);

}

p=p->pnext;

}

//не нашли фильм, сообщаем об этом пользователю

if (flag==0) printf("%s %d\n","Не найдено фильма с таким годом выпуска - ",year);

break;

}

}

}

//функция для смены местами элементов динамического массива

//работает как для смены соседних элементов так и для смены

//не соседних элементов

Film\* insInv(Film \*p1, Film \*p2)

{

Film \*pnextTemp, \*pprevTemp, \*pprev2Temp, \*pnextTemp2, \*pprevTemp2;

//здесь совершается смена указателей на объекты

//таким образом делаем перестановку в двусвязном списке

//при этом перестановка соседних элементов отличается от

//перстановки не соседних, поэтому было разработано два блока

if (p2->pprev==p1)//если элементы соседние

{

pprev2Temp=p1->pprev;

pprevTemp=p1->pprev;

pnextTemp=p2->pnext;

p1->pnext=p2->pnext;

p1->pprev=p2;

p2->pnext=p1;

p2->pprev=pprevTemp;

if (pnextTemp!=NULL) pnextTemp->pprev=p1;

if (pprev2Temp!=NULL) pprev2Temp->pnext=p2;

}

else//если элементы не соседние

{

pnextTemp=p2->pnext;

pprevTemp=p1->pprev;

pnextTemp2=p1->pnext;

pprevTemp2=p2->pprev;

p1->pnext=pnextTemp;

p1->pprev=pprevTemp2;

p2->pnext=pnextTemp2;

p2->pprev=pprevTemp;

pprevTemp2->pnext=p1;

pnextTemp2->pprev=p2;

if (pprevTemp!=NULL) pprevTemp->pnext=p2;

if (pnextTemp!=NULL) pnextTemp->pprev=p1;

}

return p2;

}

//функция для сортировки включением

void sortByFeature(Film \*p, Film \*\*begin, Film \*\*end)

{

int attr;

Film \*pnext1, \*pnext2;

printf("\nПо какому критерию выполнить сортировку?\

\n1.По ID фильма\

\n2.По названию\

\n3.По жанру\

\n4.По году выпуска\

\nВведите номер пункта:");

check(4, &attr);

//сдвигаем указатель на следующий элемент т.к. рассматривать будем всегда

//текущий элемент и элемент левее его

pnext1=p->pnext;

while (pnext1!=NULL)

{

//запоминаем элемент следующий от двух рассматриваемых

pnext2=pnext1->pnext;

//пока рассматриваемый элемент не встанет на своё место в выравненном

//в правильном порядке массиве, выполняем перестановку соседних элементов

while (1)

{

//в случае если элемент стал первым в списке или если элемент

//левее его меньше, заканчиваем делать перестановки

if (pnext1->pprev==NULL) break;

if (compare(pnext1,pnext1->pprev, attr)==1) break;

//корректируем указатель на начало или конец если необходимо

if (pnext1->pprev->pprev==NULL) \*begin=pnext1;

if (pnext1->pnext==NULL) \*end=pnext1->pprev;

//сама функция перестановки

pnext1=insInv(pnext1->pprev,pnext1);

}

//переставляем указатель на элемент следующий от двух рассматриваемых

pnext1=pnext2;

}

}

int compare(Film \*p1, Film \*p2, int attr) {

int n=0;

if (attr==1) {

if (p1->filmNmb>p2->filmNmb) n=1;

}

else if (attr==2) {

if (strcmp(p1->fname,p2->fname)>0) n=1;

}

else if (attr==3) {

if (strcmp(p1->genre,p2->genre)>0) n=1;

}

else {

if (p1->year>p2->year) n=1;

}

return n;

}

int checkLogPas(int k)

{

char login[16], pass[16], tmplogin[16], tmppass[16];

char role[5];

char strtemp1[5];

int n=0;

FILE \*pf=fopen("usr.dat","r");

if (k==1) strcpy(role,"adm");

else strcpy(role,"usr");

printf("Введите имя пользователя: ");

strcpy(login,"");

getLimStr(&login, 15, 6);

printf("Пароль: ");

strcpy(pass,"");

getLimStr(&pass, 15, 6);

if (pf!=NULL) {

fseek(pf,0,0);

while(!feof(pf)) {

strcpy(strtemp1,"");

fscanf(pf,"%4[^\r\n]\n", &strtemp1);

if (strcmp(strtemp1,role)==0) {

fscanf(pf,"%15[^\r\n]\n", &tmplogin);

fscanf(pf,"%15[^\r\n]\n", &tmppass);

if (strcmp(tmplogin,login)==0 && strcmp(tmppass,pass)==0) n++;

}

}

}

if (n!=0) return 0;

else return 1;

}

void manageUsers() {

int n, k=0;

char strtemp[5], login[15], pass[15], tmplogin[16], tmppass[16];

int flag=0;

while (flag==0) {

printf("\nУправление учётными записями пользователей\

\n1. Просмотр всех учётных записей пользователей\

\n2. Добавление новой учётной записи\

\n3. Удаление учётной записи\

\n4. Выйти в родительское меню\

\nВведите номер команды [1-4]:");

check(4, &n);

switch(n)

{

case 1: {

FILE \*pf=fopen("usr.dat","r");

int n=0, k=0;

if (pf!=NULL) {

while(!feof(pf)) {

fscanf(pf,"%4[^\r\n]\n", &strtemp);

if (strcmp(strtemp,"adm")==0 || strcmp(strtemp,"usr")==0) {

fscanf(pf,"%15[^\r\n]\n", &tmplogin);

fscanf(pf,"%15[^\r\n]\n", &tmppass);

if (k==0) {

printf("\n%2s %5s %15s %15s \n","# ","Роль ","Логин ","Пароль ");

printf("%s \n","-----------------------------------------");

k++;

}

n++;

printf("%-2d %-5s %-16s %-16s\n", n, strtemp, tmplogin, tmppass);

}

}

}

fclose(pf);

break;

}

case 2: {

int k;

char role[5], login[15], pass[15];

FILE \*pf=fopen("usr.dat","a");

fprintf(pf,"\n");

if (pf!=NULL) {

printf("\nВыберите роль добавляемого пользователя\

\n1. Администратор\

\n2. Пользователь\

\nВведите номер команды [1-2]:");

check(2, &k);

if (k==1) strcpy(role,"adm");

else strcpy(role,"usr");

fputs(role, pf);

fprintf(pf,"\n");

printf("\nВведите Имя пользователя: ");

strcpy(login,"");

getLimStr(&login, 15, 6);

fputs(login, pf);

fprintf(pf,"\n");

printf("Пароль: ");

strcpy(pass,"");

getLimStr(&pass, 15, 6);

fputs(pass, pf);

fprintf(pf,"\n");

fclose(pf);

printf("\nУчётная запись добавлена.");

}

fclose(pf);

break;

}

case 3: {

FILE \*pf;

int n=0, k=0;

int flag=0;

pf=fopen("usr.dat","r");

if (pf!=NULL) {

while(!feof(pf)) {

fscanf(pf,"%4[^\r\n]\n", &strtemp);

if (strcmp(strtemp,"adm")==0 || strcmp(strtemp,"usr")==0) {

fscanf(pf,"%15[^\r\n]\n", &tmplogin);

fscanf(pf,"%15[^\r\n]\n", &tmppass);

if (k==0) {

printf("\n%2s %5s %15s %15s \n","# ","Роль ","Логин ","Пароль ");

printf("%s \n","-----------------------------------------");

k++;

}

n++;

printf("%-2d %-5s %-16s %-16s\n", n, strtemp, tmplogin, tmppass);

}

}

}

fclose(pf);

printf("\nВведите номер пользователя в списке: ");

scanf("%d", &n);

k=1;

pf=fopen("usr.dat","r+");

if (pf!=NULL) {

while(!feof(pf) && flag!=1) {

fscanf(pf,"%4[^\r\n]\n", &strtemp);

if (strcmp(strtemp,"adm")==0 || strcmp(strtemp,"usr")==0) {

if (n==k) {

fseek(pf,-5,SEEK\_CUR);

fputs(" \n", pf);

flag=1;

}

k++;

}

}

}

fclose(pf);

break;

}

case 4: {

flag=1;

break;

}

}

}

}

void fileManager(Film \*p, char \*file, Film \*\*pend, Film \*\*pbegin) {

int k=0;

int flag=0;

while (flag==0) {

printf("\nУправление файлом базы данных\

\n1. Сохранить произведённые изменения\

\n2. Сменить файл хранения данных\

\n3. Создать файл для хранения данных\

\n4. Выйти в главное меню\

\nВведите номер команды [1-4]:");

check(4, &k);

switch(k)

{

case 1: {

saveList(p,file);

printf("Изменения сохранены в файл %s", file);

break;

}

case 2: {

FILE \*pf;

char newdir[100]={""};

printf("Введите директорию:\n");

getLimStr(&newdir, 200, 100);

pf=fopen(newdir,"r");

if (pf!=NULL) {

fclose(pf);

strcpy(file,newdir);

printf("База данных загружена.");

loadList(p,newdir,pend,pbegin);

} else printf("Ошибка загрузки файла базы данных");

fflush(stdin);

break;

}

case 3: {

FILE \*newfile;

char newdir[100]={""};

printf("Введите директорию:\n");

getLimStr(&newdir, 200, 100);

newfile=fopen(newdir,"w");

if (newfile!=NULL) {

strcpy(file,newdir);

fclose(newfile);

printf("База данных создана.");

} else printf("Ошибка создания файла для хранения данных");

break;

}

case 4: {

flag=1;

break;

}

}

}

}

void check(int k, int \*n){

int flag=0;

char r[3];

while(flag==0) {

fflush(stdin);

fgets(r,2,stdin);

\*n=atoi(r);

if (\*n>0 && \*n<k+1) {

flag=1;

} else {

printf("Введите номер команды [1-%d]:", k);

}

}

}

void getLimStr(char \*c, int l, int lim) {

int flag=0;

char \*str = (char\*) malloc((l+1)\*sizeof(char));

while (flag==0) {

fflush(stdin);

fgets(str,l,stdin);

str[strlen(str)-1]='\0';

if (strlen(str)<=lim) {

strcpy(c,str);

flag=1;

} else {

printf("Необходимо ввести заново. Количество символов не должно превышать %d:\n", lim-1);

}

}

}