

Министерство науки и высшего образования Российской Федерации
Пензенский государственный университет
Кафедра «Вычислительная техника»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту
по курсу «Программирование»
на тему «Разработка программы сложной структуры
методом нисходящего программирования»

кор.
28.12.12


Выполнил
студент группы 21BO1
Мишунин А.А.

Принял
Митрохин М. А.

Пенза 2022

Кафедра «Вычислительная техника»

« » 2022 г.

на курсовое проектирование по курсу

методом нисходящего программирования

Наименование программы: База данных «Список кино»

Название, режиссёр, жанр, кинокомпания, оценка на «Кинопоиске»

- Создание новой базы данных.
- Добавление/удаление записей, их редактирование.
- Поиск и сортировка записей.
- Сохранение базы данных в файл и удаление этого файла.
- Загрузка ранее сохраненной базы данных из файла.

1. Многомодульность.
2. Использование сложных типов данных — списков, структур, файлов.
3. Режим работы видеосистемы — текстовый/графический.
4. Устройство ввода информации — клавиатура, мышь.
5. Пользовательский интерфейс должен быть построен на основе меню и панели инструментов.
6. Наличие заставки.
7. Операционная система — Windows.
8. Система программирования — C/C++ на базе среды Microsoft Visual Studio 2019.
9. Язык программирования — Си.

Объем работы по курсу

1. Расчетная часть

Разработка программы.

2. Графическая часть

Структура модулей программы, схема алгоритма программы.

3. Экспериментальная часть

Отладка программы.

Срок выполнения проекта по разделам

1. В соответствии с графиком выполнения курсового проекта.

2.

3.

Руководитель

Задание получил

Студент

Дата выдачи задания

Дата защиты проекта

« 1 » октября

« »

« 1 »

октябрь

Введение ...

1 Постановка

1.1 Многомо

1.2 Устройс

1.3 Наличие

2 Выбор рез

2.1 Определ

2.2 Определ

2.3 Описани

3 Отладка и

4 Описание

4.1 Разработ

4.2 Разработ

5 Руководст

Заключение

Список ист

Приложени

Приложени

Приложени

Приложени

Содержание

Введение.....	5
1 Постановка задачи.....	6
1.1 Многомодульность программы.	6
1.2 Устройство ввода информации – клавиатура.....	6
1.3 Наличие заставки.....	6
2 Выбор решения.....	7
2.1 Определение необходимых модулей программы.	7
2.2 Определение структуры файла базы данных.	7
2.3 Описание разработки программы.....	8
3 Отладка и тестирование.....	9
4 Описание программы.....	12
4.1 Разработка приложения CosoleApplication3.exe	12
4.2 Разработка меню.....	17
5 Руководство пользователя.....	18
Заключение	23
Список использованных источников	24
Приложение А Исходный код программы	25
ПриложениеА.1 Файл «CensoleApplication3.cpp».....	25
ПриложениеА.2 Файл «FWork.h».....	26
ПриложениеА.3 Файл «Source.cpp»	26

Введение

Компьютеры были созданы для решения вычислительных задач, однако со временем они все чаще стали использоваться для построения систем обработки документов, а точнее, содержащейся в них информации. Такие системы обычно и называют информационными. В качестве примера можно привести систему учета отработанного времени работниками предприятия и расчета заработной платы, систему учета продукции на складе, систему учета книг в библиотеке и т.д. Все вышеперечисленные системы имеют следующие особенности:

- для обеспечения их работы нужны сравнительно низкие вычислительные мощности
- данные, которые они используют, имеют сложную структуру
- необходимы средства сохранения данных между последовательными запусками системы

Другими словами, информационная система требует создания в памяти ЭВМ *динамически обновляемой* модели внешнего мира с использованием единого хранилища - **базы данных**. Для дальнейшего обсуждения нам необходимо ввести понятие предметной области:

Предметная область - часть реального мира, подлежащая изучению с целью организации управления и, в конечном счете, автоматизации. Предметная область представляется множеством *фрагментов*, например, предприятие - цехами, дирекцией, бухгалтерией и т.д. Каждый фрагмент предметной области характеризуется множеством *объектов* и *процессов*, использующих объекты, а также множеством *пользователей*, характеризующихся различными взглядами на предметную область.

Словосочетание "динамически обновляемая" означает, что соответствие базы данных текущему состоянию предметной области обеспечивается не периодически, а в режиме реального времени. При этом одни и те же данные могут быть по-разному представлены в соответствии с потребностями различных групп пользователей.

Создание первых баз данных и СУБД стало возможно лишь с появлением достаточно дешевых и производительных устройств внешней памяти, какими стали жесткие диски (винчестеры), появившиеся во второй половине 60-х годов. В 70-е годы шла интенсивная разработка теоретических вопросов построения баз данных. В результате в начале 80-х годов на рынке появились мощные инструментальные средства проектирования и построения информационных систем. Однако, развитие информационных технологий в 90-х привело к появлению новых, более широких требований к обработке и представлению данных. Таким образом, теория баз данных, хотя и располагает впечатляющими достижениями, еще далека от завершения.

1 Постановка задачи

Необходимо разработать программу – базу данных «Список кино». Программа содержит следующие поля: Название, режиссёр, жанр, кинокомпания, оценка на “Кинопоиске”. Программа должна позволить создать и сохранить базу данных, добавлять и удалять записи, осуществлять поиск и сортировку элементов, просмотр бд. Для реализации этих требований нужно разработать интуитивно понятный интерфейс пользователя и определить названия пунктов в меню. Необходимо разработать алгоритм функционирования меню и обработки событий клавиатуры в консоли. Важно продумать структуру работы программы и механизм взаимодействия входящих в нее функций.

1.1 Многомодульность программы.

Необходимо поделить программу на логические модули. Это повысит расширяемость, упростит возможность редактирования программы и исправления возникающих ошибок при отладке или тестировании программы.

Использование сложных типов данных – односвязных списков, структур, файлов. Необходимо изучить базовые и сложные типы данных. Это необходимо для структуризации программы. Важно изучить работу с файлами. Это необходимо для возможности сохранения базы данных в файл и дальнейшего считывания из него.

Структурой в языке Си называется совокупность логически связанных переменных различных типов, сгруппированных под одним именем для удобства дальнейшей обработки. Структура – это способ связать воедино данные разных типов и создать пользовательский тип данных. Структура – тип данных, задаваемый пользователем.

Программа должна работать в текстовом или графическом режиме. Необходимо определиться с типом интерфейса и механизмом его работы, изучить приемы реализации интерфейса и разработать алгоритм работы меню пользователя.

1.2 Устройство ввода информации – клавиатура.

Необходимо научиться анализировать события, возникающие от работы с клавиатурой или мышью. Необходимо однозначно идентифицировать и выполнять те или иные действия в зависимости от действий пользователя.

1.3 Наличие заставки.

Приветствие необходимо для сообщения пользователю достаточной визуальной информации о программе и возможности перехода к работе с программой.

2 Выбор решения

2.1 Определение необходимых модулей программы.

Разработанная программа состоит из 3 модулей:

1. ConsoleApplication3.cpp
2. FWork.h
3. Source.cpp

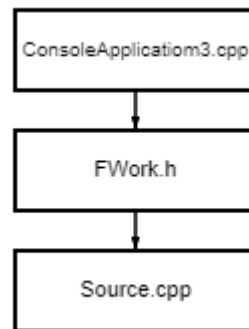


Рисунок 1 — Схема модулей программы

Заголовочный файл FWork.h необходима для объявления глобальных переменных и функций. Данная библиотека была создана специально для данной курсовой работы и представляет собой набор всех необходимых функций и переменных.

Модуль ConsoleApplication3.cpp - основной модуль программы. В нем записано несколько системных команд. Так же этот модуль содержит в себе вызов заставки и меню.

Модуль Source.cpp содержит в себе основные функции, которые нужны для работы с односвязным списком.

2.2 Определение структуры файла базы данных.

Для структурирования данных внутри файла была выбрана следующая структура данных:

```
struct cont {  
    char WName[50];  
    char Re[50];  
    char Chaner[50];  
    char Komp[50];  
    char Ocen[50];  
    int Progres;  
    char Import[20];  
};
```

База данных реализована с помощью динамического массива. Программа может делать следующее:

- 1) Создавать базу данных
- 2) Сохранять базу данных
- 3) Добавлять в базу данных записи
- 4) Удалять записи из базы данных
- 5) Осуществлять поиск по базе данных
- 6) Производить сортировку по базе данных
- 7) Выводить на экран всю базу данных
- 8) Считывать базу данных из файла

2.3 Описание разработки программы

Для написания данной программы будет использован язык программирования Си. В качестве среды программирования был выбран программный продукт Microsoft Visual Studio 2019.

При запуске программы на экран выводится заставка-приветствие. Через некоторое время перед пользователем появляется первоначальное меню, в котором он может выбрать пункт для выполнения программой соответствующего действия. Для выбора пункта меню пользователь должен ввести ту цифру, которая привязана к одному из пунктов. Для того, чтобы произвести какие-либо действия над базой данных, необходимо сначала создать её, либо загрузить из файла. Каждый из пунктов, отвечает за конкретное действие. Например, первый пункт отвечает за вывод базы данных.

Сначала нужно было написать основное меню и дополнительное меню. После этого написать функции, которые выполняют действия, указанные в пунктах меню, в дальнейшем связать эти функции с меню.

Блок-схема функции main представлена на рисунке. Она отвечает за вызов функции заставки и вызов основного меню.

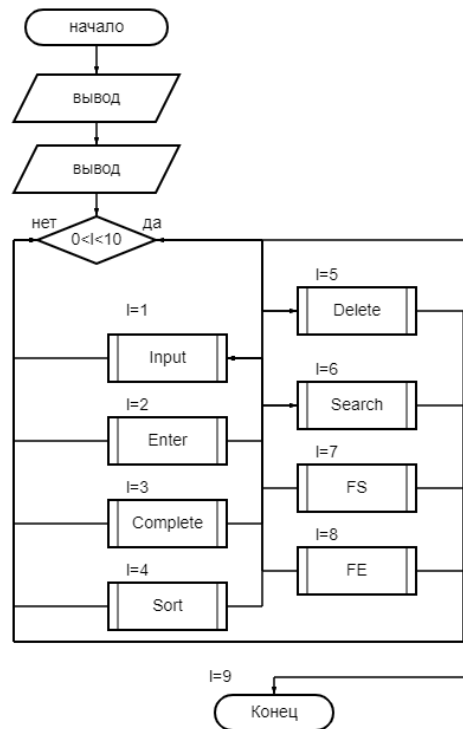


Рисунок 2 — Блок-схема функции main()

Для того чтобы, вызывать нужные функции, в функцию main была добавлена переменная L, в зависимости от её значения программа будет вызывать нужные функции. В первоначальном меню 9 пунктов, поэтому вызов функций возможен только при условии, что L будет больше 0 и меньше 10. В противном случае снова будет вызвано меню.

3 Отладка и тестирование

В качестве среды разработки была выбрана программа Visual Studio 2019. Программа предоставляет все средства необходимые при разработке и отладке разрабатываемых модулей и программ. Для отладки использовались такие инструменты как точка останова, выполнение кода по шагам, анализ содержимого локальных и глобальных переменных, анализ содержимого памяти.

Тестирование проводилось в рабочем порядке, в процессе разработки, после завершения написания программы. В ходе тестирования было выявлено и исправлено множество проблем, связанных с обработкой структуры, системой поиска и динамикой.

Тест 1: некорректный ввод

Вводимые данные: Буквы ,символы ,цифры, которые не соответствуют меню выбора.

Результат должен быть заново набор чисел в меню

```

База данных:Список кино

Введите ниже цифру
1.Вывести список фильмов
2.Дополнить список фильмов
3.Просмотренно
4.Сортировка списка
5.Удаление фильма из списка
6.Поиск фильма
7.Сохранение в файл
8.Загрузка из файла
9.Выход
0

Введите ниже цифру
1.Вывести список фильмов
2.Дополнить список фильмов
3.Просмотренно
4.Сортировка списка
5.Удаление фильма из списка
6.Поиск фильма
7.Сохранение в файл
8.Загрузка из файла
9.Выход
a

Введите ниже цифру
1.Вывести список фильмов
2.Дополнить список фильмов
3.Просмотренно
4.Сортировка списка
5.Удаление фильма из списка
6.Поиск фильма
7.Сохранение в файл
8.Загрузка из файла
9.Выход
=

Введите ниже цифру
1.Вывести список фильмов
2.Дополнить список фильмов
3.Просмотренно
4.Сортировка списка
5.Удаление фильма из списка
6.Поиск фильма
7.Сохранение в файл
8.Загрузка из файла
9.Выход

```

Рисунок 3 — Результат работы программы

Результат соответствует ожиданиям.

Тест 2. Проверка некорректного ввода оценки.

Вводимые данные:Символы, которые не соответствуют условию оценки

В результате программа должна выводить ошибку

```

Введите ниже цифру
1.Вывести список фильмов
2.Дополнить список фильмов
3.Просмотренно
4.Сортировка списка
5.Удаление фильма из списка
6.Поиск фильма
7.Сохранение в файл
8.Загрузка из файла
9.Выход
2
Введите Название фильма
ad
режисёр фильма:
da
Жанр фильма:
fd
кинокомпания, профинансировавшая фильм:
sd
оценка 'Кинопоиск':
sa
Ошибка ввода оценки

```

Рисунок 4 — Результат работы программы

Программа соответствует ожидаемым результатам

Тест 3: удаление пустого списка.

Вводимые данные: Номер удаляемого фильма.

Ожидается вывод сообщения об ошибке.

```

База данных:Список кино
Введите ниже цифру
1.Вывести список фильмов
2.Дополнить список фильмов
3.Просмотренно
4.Сортировка списка
5.Удаление фильма из списка
6.Поиск фильма
7.Сохранение в файл
8.Загрузка из файла
9.Выход
5
Введите номер удаляемого фильма:
1
Список пуст

```

Рисунок 5 — Результат работы программы

Программа соответствует ожидаемым результатам

4 Описание программы

4.1 Разработка приложения ConsoleApplication3.exe

Разработанная программа состоит из нескольких модулей. Такое разбиение позволяет более эффективно работать с кодом программы, быстрее находить и исправлять возникшие ошибки. При запуске программы на экран выводится заставка, после чего программа готова к выполнению своих функций. В зависимости от нажатой клавиши, передается управление необходимой функции. Описание состояний программы (в первоначальном меню):

Таблица 1 — Описание состояний программы в основном меню

Клавиша	Действие пользователя	Действие программы
1	Выбран пункт «Вывести список фильмов»	Программа вызывает функцию Input();
2	Выбран пункт «Дополнить список фильмов»	Программа вызывает функцию Enter();
3	Выбран пункт «Просмотрено»	Программа вызывает функцию Complete();
4	Выбран пункт «Сортировка списка»	Программа вызывает функцию Sort();
5	Выбран пункт «Удаление фильма из списка»	Программа вызывает функцию Delete();
6	Выбран пункт «Поиск фильма»	Программа вызывает функцию Search();
7	Выбран пункт «Сохранение в файл»	Программа вызывает функцию FileSave();
8	Выбран пункт «Загрузка из файла»	Программа вызывает функцию FileE();
9	Выбран пункт «Выход»	Программа закрывается

Функция Input предназначена для вывода списка фильмов. В функции реализован цикл, который отвечает за последовательный вывод пунктов списка.

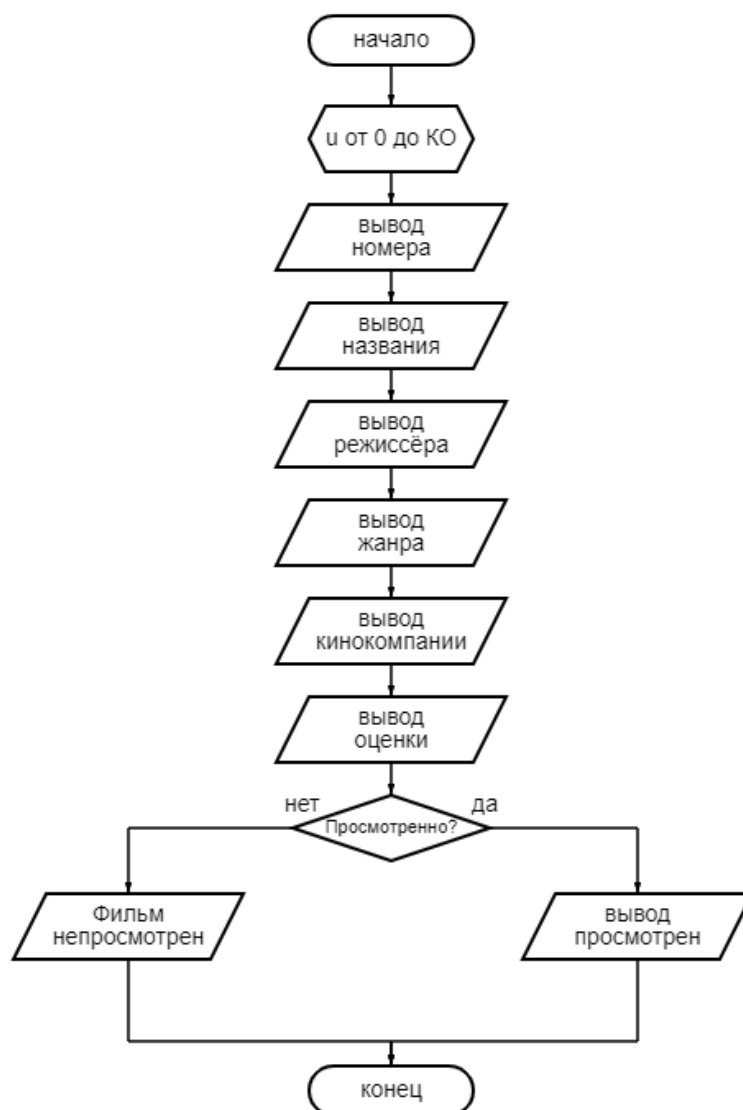


Рисунок 6 — Блок-схема функции Input()

Функция Enter предназначена для добавления в список фильм. В функции реализован ввод основных данных о новом деле, а также проверка введенных данных и добавление в динамический массив.



Рисунок 7 — Блок-схема функции Enter()

Функция Complete предназначена для изменения статуса «не просмотрено» на «просмотрено». В функции реализовано изменение статуса фильма.

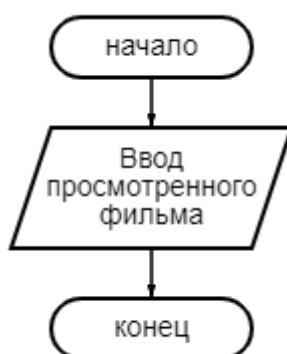


Рисунок 8 — Блок-схема функции Complete()

В зависимости от выбранного пункта список сортируется по соответствующему параметрам оценки и просмотру.

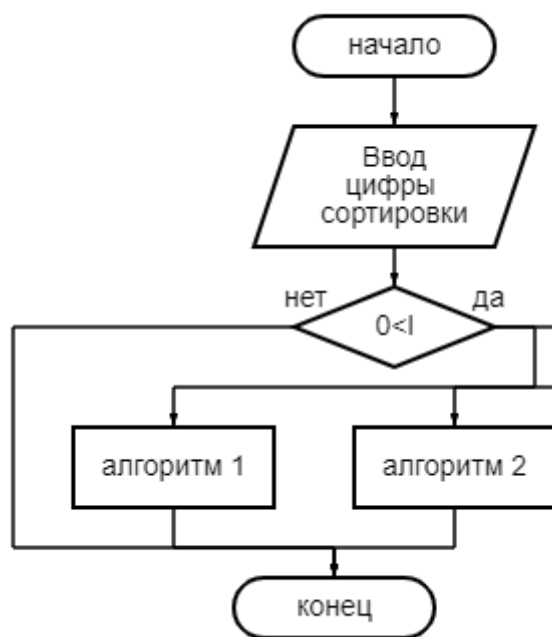


Рисунок 9 — Блок-схема функции Sort()

Функция Delete предназначена для удаления фильма из списка .



Рисунок 10 — Блок-схема функции Delete()

Функция FS предназначена для сохранения списка и количества фильмов в файл.



Рисунок 11 — Блок-схема функции FS

Если пользователь выбрал пункт «Сортировка списка», то программа вызовет дополнительное меню.

Таблица 2 — Описание состояний программы в доп. Меню

Клавиша	Действие пользователя	Действие программы
1	Выбран пункт «Сортировка по оценке»	Программа выполняет алгоритм 1
2	Выбран пункт «Сортировка по просмотру»	Программа выполняет алгоритм 2

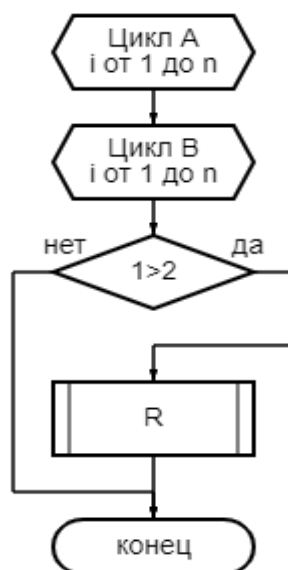


Рисунок 12 — Блок-схема алгоритма 1



Рисунок 12 — Блок-схема алгоритма 2

4.2 Разработка меню

Основное меню реализовано в файле `ConsoleApplication3.cpp`. Основное меню включает 9 пунктов:

- 1) Вывести список фильмов
- 2) Дополнить список фильмов
- 3) просмотрено
- 4) Сортировка списка
- 5) Удаление фильма из списка
- 6) Поиск фильма
- 7) Сохранение в файл
- 8) Загрузка из файла
- 9) Выход

Дополнительное меню реализовано в файле `Sourse.cpp`. Дополнительное меню включает 3 пунктов:

- 1) Сортировка по оценке
- 2) Сортировка по просмотренному

Пользователь вводит цифру, указанную в меню, и программа вызывает нужную функцию.

5 Руководство пользователя

Программа предназначена для создания списка кино. В таблице имеются такие поля, как названия фильма, режиссёра, жанра, кинокомпании и оценки. Программа может создавать базу данных, добавлять записи в нее и удалять их, осуществлять поиск и сортировку элементов, записывать и считывать базу данных в файл.

При запуске программы на экран выводится интерфейс. После чего на экране появляется основное меню.

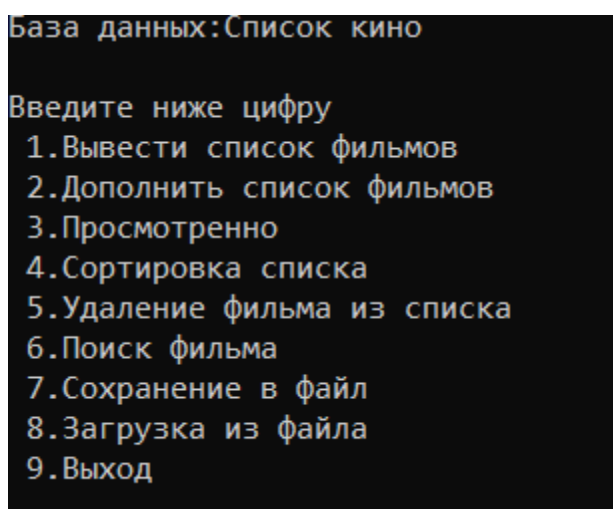


Рисунок 13 — Основное меню

Для добавления фильма в меню нужно выбрать второй пункт.

```
База данных:Список кино
Введите ниже цифру
1.Вывести список фильмов
2.Дополнить список фильмов
3.Просмотренно
4.Сортировка списка
5.Удаление фильма из списка
6.Поиск фильма
7.Сохранение в файл
8.Загрузка из файла
9.Выход
2
Введите Название фильма
```

Рисунок 14 — Создание списка фильмов

Для вывода списка фильмов нужно выбрать первый пункт

```
#1
Название: Васаби
режиссёр: Жерар Кравчик
Жанр: Боевик, комедия
кинокомпания: ЕуропаСорре
оценка 'Кинопоиск': 7.7
фильм: непросмотрен
Введите ниже цифру
1.Вывести список фильмов
2.Дополнить список фильмов
3.Просмотренно
4.Сортировка списка
5.Удаление фильма из списка
6.Поиск фильма
7.Сохранение в файл
8.Загрузка из файла
9.Выход
```

Рисунок 15 — Вывод списка дел

Пользователь может пометить фильм просмотренным , выбрав третий пункт меню.

```
Введите ниже цифру
1.Вывести список фильмов
2.Дополнить список фильмов
3.Просмотренно
4.Сортировка списка
5.Удаление фильма из списка
6.Поиск фильма
7.Сохранение в файл
8.Загрузка из файла
9.Выход
3
Введите номер номер просмотренного фильма:
1
```

Рисунок 16 — Просмотрен фильм

Пользователь может сортировать список.

```
Введите ниже цифру
1.Вывести список фильмов
2.Дополнить список фильмов
3.Просмотренно
4.Сортировка списка
5.Удаление фильма из списка
6.Поиск фильма
7.Сохранение в файл
8.Загрузка из файла
9.Выход
4

Введите цифру
1.Сортировка по оценке
2.Сортировка по просмотру
```

Рисунок 17 — сортировка списка фильмов

Пользователь может удалить фильм.

```

Введите ниже цифру
1.Вывести список фильмов
2.Дополнить список фильмов
3.Просмотренно
4.Сортировка списка
5.Удаление фильма из списка
6.Поиск фильма
7.Сохранение в файл
8.Загрузка из файла
9.Выход
5
Введите номер удаляемого фильма:

```

Рисунок 18 — удаление из списка фильм

Пользователь может найти фильм по названию.

```

Введите ниже цифру
1.Вывести список фильмов
2.Дополнить список фильмов
3.Просмотренно
4.Сортировка списка
5.Удаление фильма из списка
6.Поиск фильма
7.Сохранение в файл
8.Загрузка из файла
9.Выход
6
Введите название искомого фильма
Васаби

#1
Название: Васаби
режиссёр: Жерар Кравчик
Жанр: Боевик, комедия
кинокомпания: ЕигораСогр
оценка 'Кинопоиск': 7.7
фильм: непросмотрен

```

Рисунок 19 — Поиск фильма

Пользователь может сохранить список в файл.

```

Введите ниже цифру
1.Вывести список фильмов
2.Дополнить список фильмов
3.Просмотренно
4.Сортировка списка
5.Удаление фильма из списка
6.Поиск фильма
7.Сохранение в файл
8.Загрузка из файла
9.Выход
7

```

Рисунок 20 — Сохранения списка

Пользователь может загрузить список из файла.

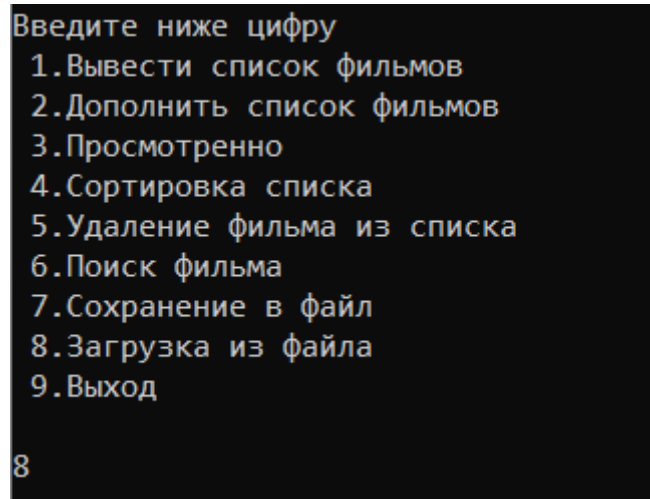


Рисунок 21 — Загрузка списка

Пользователь может выйти по цифре 9

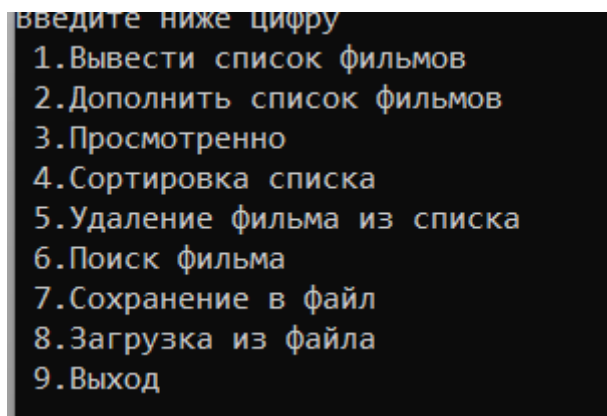


Рисунок 22 — Выход

Заключение

При выполнении данной курсовой работы были получены навыки разработки многомодульных программ. Были освоены приемы создания пользовательского интерфейса, изучены функции работы с консолью, способы обработки событий с клавиатуры. Усвоены механизмы реализации меню. Изучены функции работы с файлами. Получены базовые навыки программирования на языке Си. Изучены основные возможности среды программирования Microsoft Visual Studio 2019. Получены навыки отладки и тестирования программ.

В процессе выполнения курсовой работы была написана база данных. Программа представляет небольшой, но достаточный список возможностей.

Список использованных источников

1. <https://www.cyberforum.ru/>
2. <https://mipt.ru/dnbic/content/db.pdf>- Базы данных. Учебное пособие. – Московский государственный институт электроники и математики (Технический университет). – М., 2009.
3. <https://learn.microsoft.com/ru-ru/cpp/>

Приложение А

Исходный код программы

Приложение А.1

Файл «CensoleApplication3.cpp»

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <conio.h>
#include <windows.h>
#include <assert.h>
#include <time.h>
#include <math.h>
#include <locale>
#include "FWork.h"
#pragma warning(disable : 4996)
#pragma warning(disable : 4700)

int main(int argc, char* argv[])
{
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);

    char N = 0, l = 0;
    int L = 0;
    printf("База данных:Список кино \n");

    do {
        if (argc <= 1)
        {
            fprintf(stderr, "Аргумент командной строки не задан \n");
            system("pause");
            return 0;
        }
        printf("\nВведите ниже цифру \n 1.Вывести список фильмов \n 2.Дополнить список\n фильмов \n 3.Просмотрено \n 4.Сортировка списка \n 5.Удаление фильма из списка \n 6.Поиск\n фильма \n 7.Сохранение в файл \n 8.Загрузка из файла \n 9.Выход \n");
        scanf("%c", &l);
        while (getchar() != '\n') {}

        if (l == '.' || l == '*' || l == '/' || l == '-' || l == '_' || ((l >= 'a' && l <= 'z') || (l >= 'A' && l <= 'Z'))) {
            fprintf(stderr, "Ошибка ввода \n");
            continue;
        }
        L = l - '0';
        switch (L)
        {
            case 1:
                Input();
                break;
            case 2:
                Enter();
                break;
            case 3:
                Complete();
                break;
            case 4:
                Sort();
                break;
        }
    } while (L != 9);
}
```

```

        break;
    case 5:
        Delete();
        break;
    case 6:
        Search();
        break;
    case 7:
        FS(argv[1]);
        break;
    case 8:
        FE(argv[1]);
        break;
    case 9:
        return 0;
    }
} while (1);

```

}} Приложение А.2 Файл «FWork.h»

```

#pragma once

void Search();
void FS(char* N);
void FE(char* N);
void Input();
void Enter();
void Delete();
void Rev(int i);
void Sort();
void Complete();

```

Приложение А.3 Файл «Source.cpp»

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <conio.h>
#include <windows.h>
#include <assert.h>
#include <time.h>
#include <math.h>
#include <locale>
#include "FWork.h"
#pragma warning(disable : 4996)
#pragma warning(disable : 4700)

struct cont {
    char WName[50];
    char Re[50];
    char Chaner[50];
    char Komp[50];
    char Ocen[50];
    int Progres;
    char Import[20];
};

struct cont kn[15];
int KO = 0;
void FS(char* N)

```

```

{
    FILE* F = fopen("N", "wb");
    FILE* F1 = fopen("KO", "wb");
    fwrite(kn, sizeof(kn[0]), KO, F);
    fwrite(&KO, sizeof(KO), 1, F1);
    fclose(F);
    fclose(F1);
}
void FE(char* N)
{
    FILE* F = fopen("N", "r");
    FILE* F1 = fopen("KO", "r");
    if (!F) {
        fprintf(stderr, "Файл не существует \n");
        return;
    }
    if (!F1) {
        fprintf(stderr, "Файл не существует \n");
        return;
    }
    fread(kn, sizeof(kn[0]), 15, F);
    fread(&KO, sizeof(KO), 1, F1);

    fclose(F);
    fclose(F1);
}
void Input()
{
    for (int u = 0; u < KO; u++)
    {
        printf("\n #%d", u + 1);
        printf("\nНазвание: %s", &kn[u].WName);
        printf("\nрежиссёр: %s", &kn[u].Re);
        printf("\nЖанр: %s", &kn[u].Import);
        printf("\nкинокомпания: %s", &kn[u].Komp);
        printf("\ноценка 'Кинопоиск': % s", &kn[u].Ocen);
        if (kn[u].Progres == 1)
            printf("\n фильм: просмотрен");
        else printf("\n фильм: непросмотрен");
    }
}
void Enter()
{
    printf("Введите Название фильма \n");
    gets_s(kn[KO].WName);
    printf("режисёр фильма: \n");
    gets_s(kn[KO].Re);
    printf("Жанр фильма: \n");
    gets_s(kn[KO].Import);
    printf("кинокомпания, профинансировавшая фильм: \n");
    gets_s(kn[KO].Komp);
    printf("оценка 'Кинопоиск': \n");
    gets_s(kn[KO].Ocen);
    for (int i = 0; i < 20; i++) {
        if (kn[KO].WName[i] == '.' || kn[KO].WName[i] == '*' || kn[KO].WName[i] == '/' ||
            kn[KO].WName[i] == '-' || kn[KO].WName[i] == '_') {
            fprintf(stderr, "Ошибка ввода названия \n");
            return;
        }
        if (kn[KO].Re[i] == '.' || kn[KO].Re[i] == '*' || kn[KO].Re[i] == '/' ||
            kn[KO].Re[i] == '-' || kn[KO].Re[i] == '_') {
            fprintf(stderr, "Ошибка ввода режисёра \n");

```

```

        return;
    }
    if (kn[KO].Import[i] == '.' || kn[KO].Import[i] == '*' || kn[KO].Import[i] ==
    '/' || kn[KO].Import[i] == '-' || kn[KO].Import[i] == '_') {
        fprintf(stderr, "Ошибка ввода жанр \n");
        return;
    }
    if (kn[KO].Komp[i] == '.' || kn[KO].Komp[i] == '*' || kn[KO].Komp[i] == '/' ||
    kn[KO].Komp[i] == '-' || kn[KO].Komp[i] == '_') {
        fprintf(stderr, "Ошибка ввода кинокомпании \n");
        return;
    }
    if (kn[KO].Ocen[i] == '*' || kn[KO].Ocen[i] == '/' || kn[KO].Ocen[i] == '-' ||
    kn[KO].Ocen[i] == '_' || ((kn[KO].Ocen[i] >= 'a' && kn[KO].Ocen[i] <= 'z') ||
    (kn[KO].Ocen[i] >= 'A' && kn[KO].Ocen[i] <= 'Z'))) {
        fprintf(stderr, "Ошибка ввода оценки \n");
        return;
    }
}
kn[KO].Progres = 0;
KO++;
}
void Delete()
{
    int i;
    printf("Введите номер удаляемого фильма: \n");
    scanf("%d", &i);
    i--;
    if (KO <= 0)
    {
        fprintf(stderr, "Список пуст \n");
        return;
    }
    for (i; i < KO; i++)
    {
        strcpy(kn[i].WName, kn[i + 1].WName);
        strcpy(kn[i].Re, kn[i + 1].Re);
        kn[i].Progres = kn[i + 1].Progres;
        strcpy(kn[i].Import, kn[i + 1].Import);
    }
    KO--;
    cont* p = (cont*)malloc((KO * sizeof(cont*)));
    p = kn;
}
void R(int i)
{
    char NWName[50];
    char NRe[12];
    int NProgres;
    char NOcen[20];
    strcpy(NRe, kn[i].Re);
    strcpy(kn[i].Re, kn[i + 1].Re);
    strcpy(kn[i + 1].Re, NRe);

    strcpy(NWName, kn[i].WName);
    strcpy(kn[i].WName, kn[i + 1].WName);
    strcpy(kn[i + 1].WName, NWName);

    NProgres = kn[i].Progres;
    kn[i].Progres = kn[i + 1].Progres;
    kn[i + 1].Progres = NProgres;

```

```

strcpy(N0cen, kn[i].Ocen);
strcpy(kn[i].Ocen, kn[i + 1].Ocen);
strcpy(kn[i + 1].Ocen, N0cen);
}
void Search()
{
    cont 1;
    printf("Введите название искомого фильма \n");
    gets_s(l.WName, 50);

    for (int i = 0; i <= K0; i++)
    {
        if (strcmp(kn[i].WName, l.WName) == 0)
        {
            printf("\n #%d", i + 1);
            printf("\nНазвание: %s", &kn[i].WName);
            printf("\nрежиссёр: %s", &kn[i].Re);
            printf("\nЖанр: %s", &kn[i].Import);
            printf("\nкинокомпания: %s", &kn[i].Komp);
            printf("\nпоценка 'Кинопоиск': %s", &kn[i].Ocen);
            if (kn[i].Progres == 1)
                printf("\n фильм: просмотрен");
            else printf("\n фильм: не просмотрен");
            return;
        }
    }
    printf("\n Фильм не найдено \n");
}
void Sort()
{
    char l = 0;
    int L = 0;
    char N0cen[20];
    printf("\nВведите цифру \n 1.Сортировка по оценке \n 2.Сортировка по просмотру \n");
    scanf("%c", &l);
    getchar();
    if (l == '.' || l == '*' || l == '/' || l == '-' || l == '_' || ((l >= 'a' && l <= 'z') || (l >= 'A' && l <= 'Z'))) {
        fprintf(stderr, "Ошибка ввода \n");
    }
    L = l - '0';
    switch (L)
    {
    case 1:
        for (int j = 0; j < K0; j++)
        {
            for (int i = 0; i < K0; i++)
            {
                if (atoi(kn[i].Ocen) < atoi(kn[i + 1].Ocen) && i + 1 < K0)
                {
                    R(i);
                }
            }
        }
        break;
    case 2:
        for (int j = 0; j < K0; j++)
        {
            for (int i = 0; i < K0; i++)
            {
                if (kn[i + 1].Progres == 1)

```

```

        {
            R(i);
        }
    }
    break;
}
}
void Complete()
{
    int i, c = 0;
    printf("Введите номер просмотренного фильма: \n");
    scanf("%d", &i);
    i--;
    kn[i].Progres = 1;
    while (getchar() != '\n') {}
}

```