Министерство образования РБ

Полоцкий государственный университет

Факультет информационных технологий

Кафедра технологий программирования

**КУРСОВАЯ РАБОТА**

по дисциплине

«Основы алгоритмизации и программирования»

на тему: «Однофайловая база данных «Мультфильмы»

Выполнил: студент гр. 16-ИТ-3

Рудь В.В.

Руководитель: ассистент кафедры ТП

Магеров В.В.

Дата защиты:\_\_\_\_\_\_\_\_\_\_\_\_

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Полоцк, 2017г

**СОДЕРЖАНИЕ**

Введение

1 Анализ задания и постановка задач

2 Теоретические сведения

3 Проектирование программы

4 Реализация программы

5 Методика и результаты тестирования

Заключение

Список литературы

Приложение А

Приложение Б

Приложение В

**ВВЕДЕНИЕ**

Курсовая работа – это маленькое научное исследование, которое предполагает самостоятельность его выполнения и индивидуальный подход к подготовке содержания, грамотности оформления. Преподаватель дает задание по выполнению курсовой работы, чтобы выявить наличие знаний по предмету. После выполнения курсовая проверяется преподавателем и в случае обнаружения ошибок, отправляется на доработку. Готовая курсовая работа подлежит защите на комиссии, где с помощью тезисов излагаются основные теоретические понятия по выбранной теме, приводится пример проведенного анализа, и рекомендуются пути развития.

Целью курсового проекта является:

* Проверка знаний, полученных студентом в ходе изучения предмета.
* Закрепление и систематизация теоретических знаний с умением использовать их на практике.
* Формирование умений самостоятельной работы с литературой.
* Умение сопоставлять мнения отечественных и зарубежных авторов, высказывая при этом свои суждения.
* Формирование умений краткого изложения материала с использованием индивидуального стиля.
* Контроль полученных знаний.

Основным приоритетом курсового проекта является постановка цели, которая придаст замысел курсовой работе и определит дальнейшие этапы. Кроме того, выбранная цель будет влиять на содержание курсовой. Поэтому крайне важно уметь формулировать четко цель, поскольку именно она станет основополагающим фактором при написании введения, заключения и основных разделов работы.

Чтобы правильно определить цель выполнения задания, необходимо собрать необходимую литературу по заданной теме, чтобы разобраться в основах, тем более, если тем не известна студенту заранее. Постановка цели выполнения работы невозможно, если человек не разбирается в основных понятиях. Как можно определить, что делать, если изначально неизвестно, зачем это нужно? Поэтому, следует обратиться в источникам информации, чтобы разобраться в проблеме, подумать, какие шаги можно предпринять для ее решения. Полученный ответ на вопрос, что сделать для решения поставленной проблемы и будет целью курсовой, исходя из нее, будут определяться задачи выполнения работы и, собственно, построятся основные разделы.

Определение цели работы – является первым шагом, потому как нужно изначально знать, для чего выполняется задание, какие вопросы необходимо решить и какие меры предпринять, чтобы минимизировать негативные последствия. Цель курсовой работы определяет содержание, план действий.

Цель курсовой работы указывается во введении, сразу после обоснования актуальности темы. Исход из поставленной цели формируются задачи выполнения работы. Следует отметить, что заключение будет содержать ответ на вопрос, удалось ли достигнуть цели курсовой работы и решены ли поставленные задачи.

Цель курсовой работы должна исходить из темы, например: «Влияние солнечных лучей на азоновый слой атмосферы», следовательно, цель выполнения: «Разработка плана по уменьшению вредоносного действия солнечных лучей на азоновый слой». Только правильно поставленная цель позволит составить содержательный план и выполнить курсовую работу на отлично.

**Цель изучения и задачи курса «Основы алгоритмизации и программирования»**

Овладеть знаниями: теоретические основы алгоритмизации задач и проектирования программ; основы организации вычислительных процессов на ЭВМ; приемы проектирования программ на современном языке высокого уровня.

Научиться: характеризовать исходные данные решения задач; характеризовать выходные данные решения задач; уметь анализировать решения задач, а также возможные методы решения задач и осуществлять обоснованный выбор.

Приобрести навыки: формальной математической постановки задачи; алгоритмизации задач и программирования, отладки и выполнения на ЭВМ конкретных задач с использованием современных методов программирования.

Изучаемые темы: структуры, объединения, указатели и динамическая память, объявление указателей; процедуры и функции для работы с динамической памятью; файлы: понятие файла, виды файлов, функции для работы с файлами; алгоритмы сортировки и поиска; списки с разными видами хранения.

**1 Анализ задания и постановка задач**

Мультипликационный фильм, мультфильм  — это [фильм](https://ru.wikipedia.org/wiki/%D0%A4%D0%B8%D0%BB%D1%8C%D0%BC), выполненный при помощи средств покадровой отрисовки (включая 3D-моделирование) и предназначенный для демонстрации в кинотеатре, трансляции по телевидению, просмотре на экране компьютера и других электронных устройствах.

Целью курсовой работы является создание программы, которая содержит базу данных мультфильмов.

Согласно поставленной задаче нужно реализовать интерфейс для пользователя, с помощью которого возможно хранение информации в базе данных, удаление, редактирование, сортировка и добавление новых записей.

* В требованиях к программе оговорены следующие пункты:
* запрещается использование безусловного оператора GOTO;
* при выводе информации должна производиться предварительная очистка экрана;
* в процессе описания каждой функции должен присутствовать комментарий, объясняющий назначение и принцип работы данной функции;
* данные обязательно выводить в табличном виде;
* необходимо реализовать возможность вывода результатов работы в файл, а также выгрузку из файла.

### Для классификации мультфильмов

### По способу создания:

* [Пластилиновый](https://ru.wikipedia.org/wiki/%D0%9F%D0%BB%D0%B0%D1%81%D1%82%D0%B8%D0%BB%D0%B8%D0%BD%D0%BE%D0%B2%D0%B0%D1%8F_%D0%B0%D0%BD%D0%B8%D0%BC%D0%B0%D1%86%D0%B8%D1%8F).
* [Рисованный](https://ru.wikipedia.org/wiki/%D0%A0%D0%B8%D1%81%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F_%D0%BC%D1%83%D0%BB%D1%8C%D1%82%D0%B8%D0%BF%D0%BB%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F).
* [Кукольный](https://ru.wikipedia.org/wiki/%D0%9A%D1%83%D0%BA%D0%BE%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%BC%D1%83%D0%BB%D1%8C%D1%82%D0%B8%D0%BF%D0%BB%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F).
* [Компьютерный](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D0%BA%D0%B0).
* [Песочный](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%81%D0%BE%D1%87%D0%BD%D0%B0%D1%8F_%D0%B0%D0%BD%D0%B8%D0%BC%D0%B0%D1%86%D0%B8%D1%8F).

### По продолжительности:

* [Полнометражный](https://ru.wikipedia.org/wiki/%D0%9F%D0%BB%D0%B0%D1%81%D1%82%D0%B8%D0%BB%D0%B8%D0%BD%D0%BE%D0%B2%D0%B0%D1%8F_%D0%B0%D0%BD%D0%B8%D0%BC%D0%B0%D1%86%D0%B8%D1%8F).
* [Короткометражный](https://ru.wikipedia.org/wiki/%D0%A0%D0%B8%D1%81%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F_%D0%BC%D1%83%D0%BB%D1%8C%D1%82%D0%B8%D0%BF%D0%BB%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F).

### По способу показа:

* Театральный мультфильм — такие мультфильмы сначала показывают в кинотеатрах, а позже на телевидении и видеоносителях. Сейчас обычно таким способом показывают только полнометражные мультфильмы, хотя иногда (например, у многих диснеевских мультфильмов) перед полнометражным показывают короткий мультфильм. Раньше, до широкого распространения телевизоров, в кинотеатрах показывали и короткометражные мультфильмы.
* [Direct-to-video](https://ru.wikipedia.org/wiki/Direct-to-video) — полнометражный мультфильм, который по какой-то причине не стали показывать в кинотеатре
* Короткометражные мультфильмы и [мультсериалы](https://ru.wikipedia.org/wiki/%D0%9C%D1%83%D0%BB%D1%8C%D1%82%D1%81%D0%B5%D1%80%D0%B8%D0%B0%D0%BB), которые показывают по телевизору и распространяют на видеоносителях.

В связи с тем, что данное программа должно работать с мультфильмами, важно правильно выделить главные критерии, по которым можно рассматривать каждый отдельный мультфильм.

В качестве таковых были выбраны следующие характеристики:

1. Название мультфильма (пример: Шрек 3).
2. Жанр мультфильма (пример: Мистика).
3. Год выхода мультфильма (пример: 2011).
4. Страна, создавшая мультфильм (пример: Америка)
5. Способ создания мультфильма (пример: Пластилиновый).
6. Продолжительность мультфильма (пишется в формате xx:xx:xx).
7. Способ показа мультфильма (пример: Телевидение)
8. Кассовый сбор мультфильма (пример: 150000.00).

В процессе написания программы должны быть реализованы следующие пункты:

* инициализация и удаление списка;
* добавление новых записей в начало, в конец и после указанной записи;
* вывод списка в табличном виде;
* удаление единичной записи;
* сортировка списка в любом направление по любому из полей;
* поиск записи по полному либо частичному совпадению по любому из полей;
* редактирование записи по любому полю;
* сохранение и выгрузка структуры с добавлением либо перезаписью.

**2 Теоретические сведения**

**Язык С** (читается "Си") создан в начале 70х годов, когда Кен Томпсон и Дэннис Ритчи из Bell Labs разрабатывали операционную систему UNDC. Сначала они создали часть компилятора С, затем использовали ее для компиляции остальной части компилятора С и, наконец, применили полученный в результате компилятор для компиляции UNIX. Операционная система UNIX первоначально распространялась в исходных кодах на С среди университетов и лабораторий, а получатель мог откомпилировать исходный код на С в машинный код с помощью подходящего компилятора С.

**Бьерн Страуструп** высвободил объектно-ориентированный потенциал С путем перенесения возможностей классов Simula 67 в С. Первоначально новый язык носил имя "С с классами" и только потом стал называться C++. Язык C++ достиг популярности, будучи разработанным в Bell Labs, позже он был перенесен в другие индустрии и корпорации. Сегодня это один из наиболее популярных языков программирования в мире. C++ наследует как хорошие, так и плохие стороны С.

**Бьерн Страуструп**: "Я придумал C++, записал его первоначальное определение и выполнил первую реализацию. Я выбрал и сформулировал критерии проектирования C++, разработал его основные возможности и отвечал за судьбу предложений по расширению языка в комитете по стандартизации C++", - пишет автор самого популярного языка программирования. - "Язык C++ многим обязан языку C, и язык C остается подмножеством языка C++ (но в C++ устранены несколько серьезных брешей системы типов C). Я также сохранил средства C, которые являются достаточно низкоуровневыми, чтобы справляться с самыми критическими системными задачами. Язык C, в свою очередь многим обязан своему предшественнику, BCPL; кстати, стиль комментариев // был взят в C++ из BCPL. Другим основным источником вдохновения был язык Simula67. Концепция классов (с производными классами и виртуальными функциями) была позаимствована из него. Средства перегрузки операторов и возможность помещения объявлений в любом месте, где может быть записана инструкция, напоминает Algol68. "

Название **C++** выдумал Рик Масситти. Название указывает на эволюционную природу перехода к нему от C. "++" - это операция приращения в C. Чуть более короткое имя C+ является синтаксической ошибкой; кроме того, оно уже было использовано как имя совсем другого языка. Знатоки семантики C находят, что C++ хуже, чем ++C. Названия D язык не получил, поскольку он является расширением C и в нем не делается попыток исцеляться от проблем путем выбрасывания различных особенностей...

Связный список — базовая динамическая структура данных в информатике, состоящая из [узлов](https://ru.wikipedia.org/wiki/%D0%A3%D0%B7%D0%B5%D0%BB_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)), каждый из которых содержит как собственно [данные](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%B5_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)), так и одну или две [ссылки](https://ru.wikipedia.org/wiki/%D0%A1%D1%81%D1%8B%D0%BB%D0%BA%D0%B0_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) («связки») на следующий и/или предыдущий узел списка. Принципиальным преимуществом перед [массивом](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%81%D1%81%D0%B8%D0%B2_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) является структурная гибкость: порядок элементов связного списка может не совпадать с порядком расположения элементов данных в памяти компьютера, а порядок обхода списка всегда явно задаётся его внутренними связями.

В [информатике](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0) линейный список обычно определяется как [абстрактный тип данных](https://ru.wikipedia.org/wiki/%D0%90%D0%B1%D1%81%D1%82%D1%80%D0%B0%D0%BA%D1%82%D0%BD%D1%8B%D0%B9_%D1%82%D0%B8%D0%BF_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) (АТД), формализующий понятие упорядоченной [коллекции данных](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BB%D0%BB%D0%B5%D0%BA%D1%86%D0%B8%D1%8F_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85). На практике линейные списки обычно реализуются при помощи [массивов](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%81%D1%81%D0%B8%D0%B2_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) и связных списков. Иногда термин «список» неформально используется также как синоним понятия «связный список». К примеру, АТД нетипизированного [изменяемого](https://ru.wikipedia.org/wiki/%D0%98%D0%B7%D0%BC%D0%B5%D0%BD%D1%8F%D0%B5%D0%BC%D1%8B%D0%B9_%D0%BE%D0%B1%D1%8A%D0%B5%D0%BA%D1%82) списка может быть определён как набор из [конструктора](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%82%D0%BE%D1%80_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) и основных операций:

* Операция, проверяющая список на пустоту.
* Три операции добавления объекта в список (в начало, конец или внутрь после любого (n-ого) элемента списка);
* Операция, вычисляющая первый (головной) элемент списка;
* Операция доступа к списку, состоящему из всех элементов исходного списка, кроме первого.

#### Характеристики

Перечислим основные свойства списка:

* элемент списка доступен в программе через указатель. Между указателем и элементом списка имеется такая же взаимосвязь, как между индексом в массиве и элементом массива;
* порядок следования элементов определяется последовательностью связей между элементами. Изменение порядка следования элементов (вставка, удаление) осуществляются изменением установки указателей на соседние элементы.
* логический (порядковый) номер элемента списка также задается его естественной нумерацией в цепочке элементов;
* список удобен для использования именно как динамическая структура данных: элементы списка обычно создаются как динамические переменные, а связи между ними устанавливаются программно (динамически);
* список обладает свойством локальности изменений: при вставке/удалении элемента изменения касаются только текущего и его соседей. Вспомним массив: при вставке/удалении его элементов происходит физическое перемещение (сдвиг) всех элементов от текущего до конечного.

Отсюда следует, что преимущества списков проявляются в таких структурах данных, где операции изменения порядка превалируют над операциями доступа и поиска.

Списки обладают также рядом недостатков:

* сложность определения адреса элемента по его индексу (номеру) в списке;
* на поля-указатели (указатели на следующий и предыдущий элемент) расходуется дополнительная память (в массивах, например, указатели не нужны);
* работа со списком медленнее, чем с массивами, так как к любому элементу списка можно обратиться, только пройдя все предшествующие ему элементы;
* элементы списка могут быть расположены в памяти раздельно, что окажет негативный эффект на кэширование процессора;

В двусвязном списке ссылки в каждом узле указывают на предыдущий и на последующий узел в списке. Как и односвязный список, двусвязный допускает только последовательный доступ к элементам, но при этом дает возможность перемещения в обе стороны. В этом списке проще производить удаление и перестановку элементов, так как легко доступны адреса тех элементов списка, указатели которых направлены на изменяемый элемент.

**Указатель –** этопеременная, значением которой является адрес, по которому располагаются данные.

**Типизированный указатель** – указатель, содержащий адрес данных определенного типа (системного или пользовательского).

**Нетипизированный указатель** – указатель, содержащий адрес данных неопределенного типа (просто адрес).

Классификация по области доступа определяется методом адресации принятой для семейства процессоров x86: адрес состоит из двух элементов: сегмент и смещение.

**Адрес** – это номер ячейки памяти, в которой или с которой располагаются данные.

Классифицировать указатели можно:

* по типу данных (типизированные и не типизированные указатели);
* по области доступа (ближние и дальние указатели).

**Динамическая память** – это область (блок) памяти выделенный для нужд программы в процессе работы программы (а не заранее).

Основными двумя действиями над динамической памятью являются: выделение и освобождение. В языке С функции для осуществления этих действий описаны в библиотеке **stdlib.h**.

**Структура** – это сложный тип данных представляющий собой упорядоченное в памяти множество элементов различного типа. Каждый элемент в структуре имеет свое имя и называется полем. Размер структуры определяется суммой размеров всех элементов.

**Файл** – именованная область данных на каком-либо носителе информации (жесткий диск, дискета, компакт-диск и т.д. и т.п.).

Язык С, как и другие языки программирования высокого уровня, позволяет осуществлять операции файлового ввода и вывода. Основной алгоритм обработки файлов выполняется в три действия в следующем порядке:открытие файла;чтение и/или запись данных в файл; закрытие файла.

В языке С все файлы делятся на два вида:

* бинарные;
* текстовые.

**Бинарный файл** – файл, содержащий структурированную или не структурированную информацию, представленную в двоичном (бинарном) виде.

**Текстовый файл** – файл, содержащий структурированную или не структурированную информацию, представленную в текстовом (ASCII символы) виде.

Операции чтения и записи для текстовых и бинарных файлов отличаются друг от друга (в языке С реализованы в виде различных функций). В языке С для работы с файлами набор функций реализован в библиотеке stdio.h.

**Функция** – это синтаксически выделенный именованный программный модуль, выполняющий определенное действие или группу действий. Каждая функция имеет свой интерфейс и реализацию.

**Реализация функции** – тело функции, содержащее внутренние (локальные) данные функции и программный код, выполняющий действия согласно переданным в функцию параметрам и возвращающий значение, соответствующего интерфейсу функции типа.

**3 Проектирование программы**

Так как нам необходимо создать полноценный пользовательский интерфейс, нам нужно реализовать следующий список функций:

* функция считывания данных с файла;
* функция сохранения данных в файл;
* функция вывода списка в виде таблицы;
* функция добавления записи;
* функция удаления записи;
* функция редактирования записи;
* функция перемещения записи в списке;
* функция для перехода к выбранной записи;
* функция поиска записи с учетом регистра или без;
* функция сортировки списка по любому полю в любом направлении;

При запуске программы на экране будет появляться главное меню, состоящее из 5 пунктов:

* Просмотр/редактирование.
* Загрузить из файла.
* Сохранить в файл.
* Поле с вводом имени файла.
* Выход.
* Помощь.

При переходе в первый пункт меню, происходит вывод списка в виде таблицы. Помимо вывода здесь происходит редактирование списка (добавление, удаление, перемещение, редактирование записей), сортировка и поиск записей.

При выборе второго пункта меню, происходит попытка загрузки записей из файла, указанного в четвертом пункте. В случае правильного имени файла предоставляется выбор режима загрузки (перезапись или добавление).

При выборе третьего пункта меню, происходит попытка сохранения записей в файл, указанный в четвертом пункте. В случае правильного имени файла предоставляется выбор режима сохранения (перезапись или добавление).

При выборе четвертого пункта меню, происходит ввод имени файла для дальнейшей работы с ним.

При выборе пятого пункта меню, в случае наличия списка, будет предложено его сохранение в указанный файл, иначе происходит выход из программы.

Для выбора последнего пункта требуется нажать клавишу “h” (help), далее происходит переход в меню помощь, здесь описаны навигационные и горячие клавиши для работы со списком.

**4 Реализация программы**

Для реализации программы были созданы следующие структуры данных:

* tempcstruct - содержит данные о мультфильмах;

**Листинг 4.1** – Структура tempcstruct

1: struct tempcstruct

2: {

3: char name[29];

4: char genre[25];

5: char year[5];

6: char country;

7: char wfcreating[29];

8: char time[9];

9: char mfdelivery[25];

10: double money;

11: };

* cstruct – звено списка, хранит данные и два указателя на соседние звенья;

**Листинг 4.1** – Структура tempcstruct

1: struct cstruct

2: {

3: char name[29];

4: char genre[25];

5: char year[5];

6: char country;

7: char wfcreating[29];

8: char time[9];

9: char mfdelivery[25];

10: double money;

11: cstruct \*next;

12: cstruct \*pref;

13: };

* HEAD, TAIL, n – указатели на первый и последний элементы, количество элементов;

**Листинг 4.1** – Структура tempcstruct

1: cstruct \* HEAD;

2: cstruct \* TAIL;

3: int n=0;

Для корректной работы програмы объявлены следующие функции:

Для работы со списком:

* int deleteel(int m, int &n, cstruct \*&HEAD, cstruct \*&TAIL) – удаление записи.
* cstruct \* createbase(int &n, cstruct \*&HEAD, cstruct \*&TAIL) – создание списка
* cstruct \* addelbeg(int &n, cstruct \*& HEAD, cstruct \*& TAIL) – добавление элемента в начало.
* cstruct \* addelend(int &n, cstruct \*& HEAD, cstruct \*& TAIL) – добавление элемента в конец.
* cstruct \* addeln(int &n, cstruct \*& HEAD, cstruct \*& TAIL, int m) – добавление элемнта в позицию.

Для сортировки:

* int sort(int &n, cstruct \*&HEAD, int change, int field) – сортирует поле field (1,2,3...) по возрастанию или убыванию в зависимости от переменной change.
* int sortfield(tempcstruct \*& temp, tempcstruct \*& t, int j, int d, int field) – функция сравнивает две строки, хранящиеся в полях field.
* tempcstruct \* unloadingINarray(int &n, cstruct \*&HEAD) – выгрузка списка в динамический массив.
* void unloadingFROMarray(int &n, tempcstruct \*&temp, cstruct \*&HEAD) – выгрузка из массива в список.

Для поиска:

* int find\_word(cstruct \*&TAIL,cstruct \*&HEAD, char \*& word, int &regWORD, int &n, int &tW, int &update) – поиск по слову.

Для вывода:

* int outputtable(cstruct \*&HEAD, cstruct \*&TAIL, int &n) – вывод записей в табличном виде.
* int output\_find(cstruct \*&HEAD, int i) – вывод одной записи для редактирования. Здесь передается голова и номер элемента в списке.
* void output\_record\_find(int i,int POS, cstruct \*&HEAD, int col, int bg) – вывод одной записи в таблице. Здесь передается голова и номер элемента в списке.
* void output\_record(int i, cstruct \*&p, int col, int bg) – вывод одной записи в таблице.
* int output(cstruct \*&p, int i) – вывод одной записи для редактирования.
* void printcbg(short x, short y, char chr[256], int colr, int background) – печатает указанную строку начиная с координат x,y с цветом colr и фоном background.
* void bar(int x, int y, int x1 , int y1,int col, int bg,char a[]) – рисует прямоугольник. Левая правая координата x,y и правая нижняя x1,y1 с цветом col и фоном bg, заполняя прямоугольник символом a;

Для работы с файлами:

* int save\_DB(cstruct \*&HEAD, int &n, char \*namefile) – сохранение списка в файл.
* int load\_DB(cstruct \*&HEAD, cstruct \*&TAIL, int &n, char \*namefile) – загрузка списка из файл.

Для ввода с проверкой:

* int input\_word\_time(char \* word, int gran, int Xposcursor, int Yposcursor, int &foroutput) – ввод времени.
* int input\_word\_double(char \* word, int gran, int Xposcursor, int Yposcursor, int &foroutput) – ввод числа double.
* int input\_word\_year(char \* word, int gran, int Xposcursor, int Yposcursor, int &foroutput) – ввода года.
* int input\_namefile(char \* word, int gran, int Xposcursor, int Yposcursor, int &foroutput, int xclear, int xsize) – ввод имени файла.
* int input\_word(char \* word, int gran, int Xposcursor, int Yposcursor, int &foroutput, int xclear, int xsize) – ввод слов.
* int input\_word\_search(char \*& word) – ввод слова поиска.

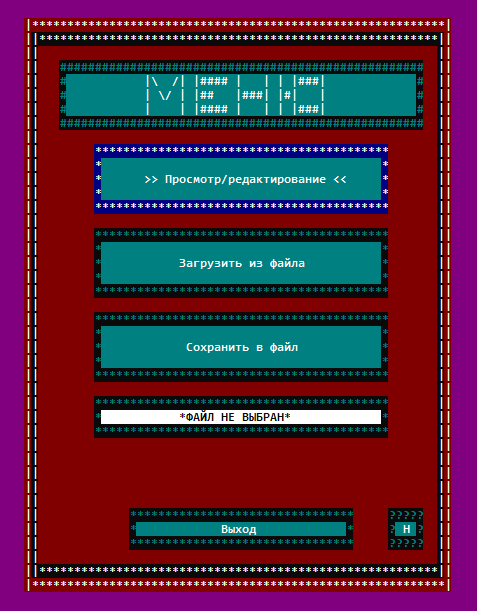
Для меню:

* int menu(char \* namefile, cstruct \*&HEAD, cstruct \*&TAIL, int &n) – главное меню.
* int help() – меню «помощь».
* int end(int &n, cstruct \*& HEAD, char \*namefile) – меню выхода.

Методом сортировки был использован метод Шелла. Алгоритм сортировки, являющийся усовершенствованным вариантом сортировки вставками. Идея метода Шелла состоит в сравнении элементов, стоящих не только рядом, но и на определённом расстоянии друг от друга. Иными словами — это сортировка вставками с предварительными «грубыми» проходами. Аналогичный метод усовершенствования пузырьковой сортировки называется сортировка расчёской. Принцип работы алгоритма Шелла состоит в том, что в исходном наборе элементов (массиве) сначала упорядочиваются элементы, расположенные на расстоянии друг от друга, т.е. итерации по массиву осуществляются через несколько элементов. Затем шаг уменьшается и производится сортировка с новым размером шага. Этот размер шага указан в массиве steps[]={9,5,3,1}, сортировка происходит в четыре этапа, начиная с шага в восемь элементов и заканчивая шагом в один элемент. Так продолжается до тех пор, пока шаг не станет равным единице. Таким образом, алгоритм Шелла представляет собой алгоритм сортировки, в котором ускорение процесса осуществляется за счет того, что на первых этапах (с большим шагом) обрабатывается меньше элементов, и они упорядочиваются быстрее, перемещаясь по массиву с большими шагами. Данный алгоритм относят к классу быстрых алгоритмов скорее условно, чем в действительности.

**5 Тестирование программы**

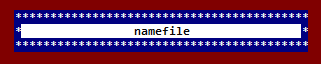
При загрузке программы пользователь наблюдает следующее меню (рисунок 5.1):



**Рисунок 5.1** – Главное меню

Для того, чтобы загрузить или сохранить список, пользователь должен ввести имя файла, для этого нужно выбрать четвертый пункт меню. (Рисунок 5.2):

C:\Users\Dell\Desktop\1.PNG



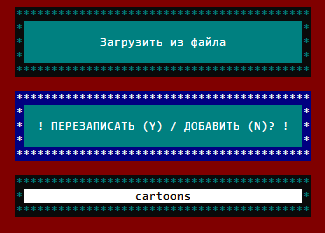
**Рисунок 5.2** – ввод имени файла

Для того, чтобы загрузить список из файла выбираем второй пункт меню. В зависимости от содержания файла, пользователь увидит соответствующее сообщение (рисунок 5.3):



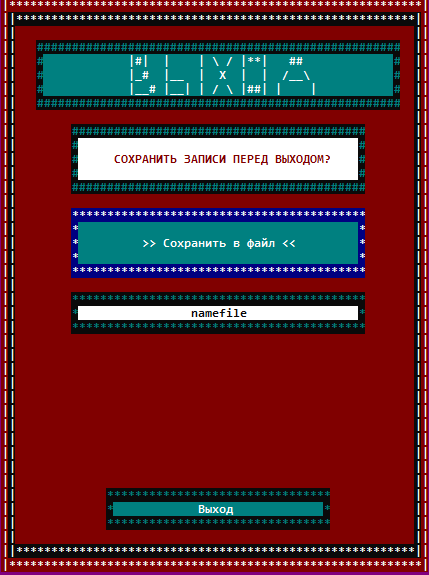
**Рисунок 5.3** – сообщение при загрузке

Чтобы сохранить имещийся список, пользователь должен выбрать третий пункт меню. В зависимости от содержания файла и от наличия списка, пользователь увидит соответствующее сообщение (Рисунок 5.4):



**Рисунок 5.4** – сообщение при сохранении

Для того чтобы выйти, выбираем пятый пункт меню. Если при выборе этого пункта имеется наличие списка, пользователь увидит следующее меню (Рисунок 5.5):



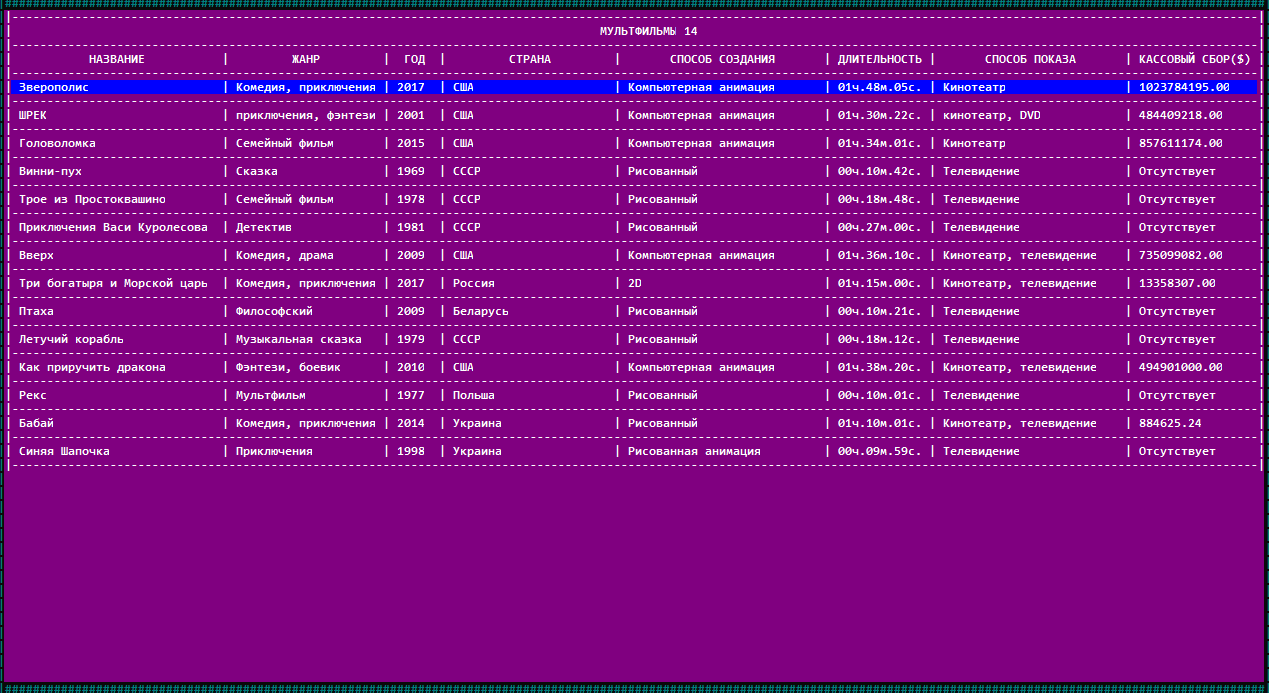
**Рисунок 5.5** – меню выхода

Для того чтобы в меню помощи пользователь должен нажать клавишу “h” (Рисунок 5.6):

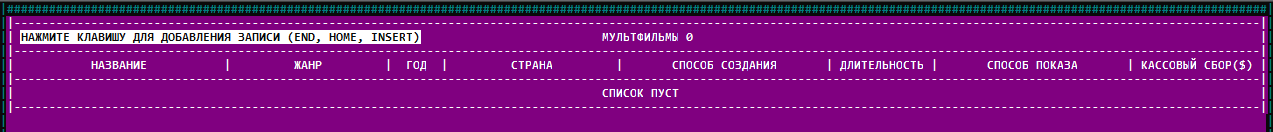


**Рисунок 5.6** – меню помощи

При выборе первого пункта меню после загрузки данных из файла, пользователь видит список в табличном виде (Рисунок 5.7) или в случае если список отсутствует пользователю будет предложено создать элемент (Рисунок 5.8) :

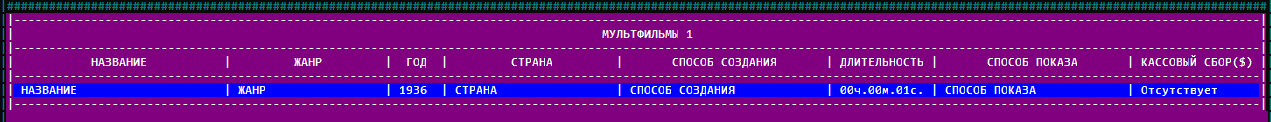


**Рисунок 5.7 –** список в табличном виде



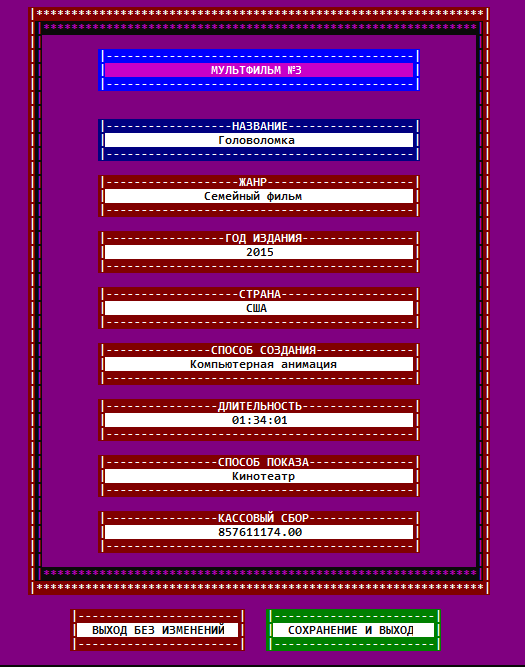
**Рисунок 5.8** – предложение создать элемент

При создании нового элемента каждое из полей изначально содержит подсказку в виде названия указанного поля (Рисунок 5.9):



**Рисунок 5.9** – новый элемент

Для того, чтобы отредактировать запись, пользователь должен выбрать нужную запись и нажать клавишу “Enter”. После происходит переход в меню редактирования.

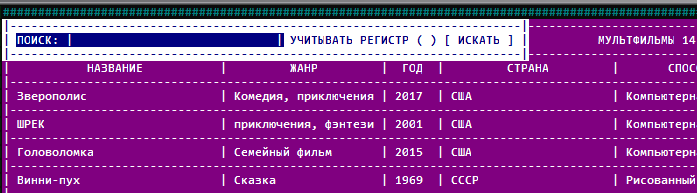


**Рисунок 5.10 –** меню редактирования

Для того чтобы отредактированить конкретное поле, пользователь должен выбрать требуемое с помощью клавиш навигаций и нажать клавишу “Enter”.

Чтобы выйти из меню редактирования без изменений или с сохранением нужно выбрать соответствующий пункт меню.

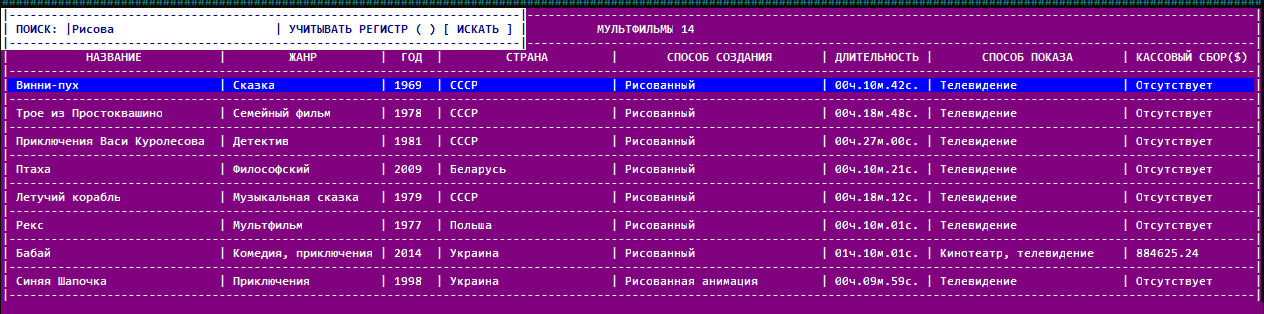
Для поиска записи пользователь должен нажать комбинацию клавиш “Shift+f”. После этого откроется меню поиска (Рисунок 5.11).



**Рисунок 5.11** – меню поиска

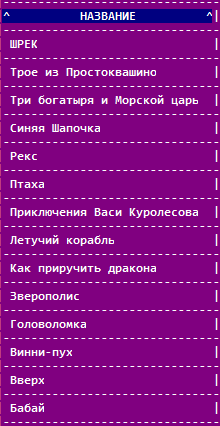
Здесь с помощью клавиш перемещения пользователь выбирает из трех пунктов. Первый пункт позволяет ввести слово для поиска, второй – режим поиска: с учетом регистра или без, третий – сам поиск. После ввода слова и выбора пункта «Искать» происходит поиск с частичным совпадением.

После пользователь может перемещаться по найденным элементам для их удаления или редактирования (Рисунок 5.12). Для того чтобы выйти из меню поиска требуется нажать клавишу «Escape».



**Рисунок 5.12** – перемещение по найденным элементам

Для того, чтобы отсортировать список по определенному полю пользователь должен с помощью навигациооных клавиш выбрать соответствующее поле и нажать клавишу «Enter». После этого произойдет сортировка по возрастанию. При этом на выбранном поле появятся стрелки в виде символа «v».Чтобы отсортировать список по убыванию пользователь должен снова нажать клавишу «Enter» на этом поле. Стрелки примут вид символа «^» (Рисунок 5.13).



**Рисунок 5.13** – сортировка по убыванию