СОдержание

Введение 2

1 Анализ литературных источников 3

1.1 Основные теоретические понятия 3

1.1.1 Динамические структуры данных 3

1.1.2 Запись 3

1.1.3 Абстрактный тип данных «список» 4

1.1.4 Типизированные файлы 4

1.1.5 Текстовые файлы 5

1.2 Анализ аналогичных программных средств 5

2 Постановка задачи 8

3 Разработка программного средства 9

3.1 Структура главного меню 10

3.2 Структура программного средства 10

3.3 Структура типов 11

3.4 Структура данных 12

3.4.1 3.4.1 Структура данных основной программы3 12

3.4.2 3.4.2 Структура данных алгоритма ChangeLV (HeaderList,ListOfFilm ) 12

3.4.3 3.4.3 Структура данных алгоритма SaveFilm (AlwaysAsk, changes, FFileName) 13

3.4.4 Структура данных алгоритма ReadFile (FFileName, ListOfFilm, HeaderList) 13

3.4.5 Структура данных алгоритма Search(StrSearch, HeaderList, ListOfFilm) 14

3.4.6 Структура данных алгоритма SortLV(HeaderList, ListOfFilm, SRav) 14

3.4.7 Структура данных алгоритма showLV (HeaderList,ListOfFilm) 15

3.4.8 Структура данных алгоритма randomFilm ( HeaderList, ListOfFilm ) 15

3.4.9 Структура данных алгоритма DeletLV(HeaderList, ListOfFilm) 15

3.5 Описание алгоритмов 16

3.6 Схема алгоритма решения задачи по ГОСТ 19.701-90 17

# Введение

Кинематограф относительно новый вид искусства. Его история в сравнении с тысячелетней историей музыки, живописи, театра очень коротка. Но это не мешает кино оставаться уже на протяжении нескольких десятков лет самым массовым видом искусства. Историю кино принято отсчитывать от первого публичного киносеанса, организованного братьями Люмьер в Париже в «Гран кафе» на бульваре Капуцинок 28 декабря 1895. Это было «Прибытие поезда на вокзал Ла Сиота». Сегодня кинематограф - прежде всего индустрия, отрасль экономики, приносящая огромную прибыль развитым государствам.

В рамках данного курсового проекта было разработано программное средство для хранения информации о фильмах, предусматривающее возможности добавления, удаления, редактирования записей и поиска данных.

Настоящая пояснительная записка содержит следующие разделы:

– введение, в разделе анализируются цели создания приложения о фильмах, кратко излагается содержания разделов пояснительной записки;

– аналитический обзор литературы, в разделе приводятся основные теоретические данные необходимые в программы, анализируются аналогичные программные средства и ставятся задачи приложения;

XX– разработка алгоритма, в разделе приведена общая схема программы и схемы двух используемых процедур;

– разработка программного средства и обоснование технических приемов программирования, в этом разделе описывается реализации программы, объясняется выбор компонентов и структуры;

– тестирование программного средства, в разделе разработаны способы проверки правильности и работоспособности отдельных функций и программной средства в целом, спланированы и проведены экспериментальные исследования, отражающие функционирование разработанного программного средства;

– руководство пользователя, раздел содержит последовательность действий, выполняемых пользователем для успешной работы системы;

– заключение, в разделе перечисляются основные результаты и итоги проекта. ХХХ

# Анализ литературных источников

## Основные теоретические понятия

Разработка программного средства для хранения информации о фильмах подразумевает работу с динамическими структурами данных, а именно с однонаправленными списками. Входные и выходные данные должны иметь формат типизированных файлов, кроме того, имеется возможность вывести информацию о занесенных фильмах в текстовый файл. Поэтому рассмотрим подробнее эти понятия.

### Динамические структуры данных

В любой современной вычислительной машине важным ресурсом является память. Поэтому управление ею – одна из важнейших задач программиста. Различают несколько основных способов управления памятью. Статическое – во время трансляции программы, что эффективно, поскольку в ходе ее выполнения на управление памятью не расходуются ресурсы системы. Альтернативным способом работы с памятью являются динамические структуры данных – это структуры данных, память под которые выделяется и освобождается по мере необходимости.

Динамические структуры данных в процессе существования в памяти могут изменять не только число составляющих их элементов, но и характер связей между элементами. При этом не учитывается изменение содержимого самих элементов данных.

Следует отчетливо понимать, что работа с динамическими данными замедляет выполнение программы, поскольку доступ к величине происходит в два шага: снача­ла ищется указатель, затем по нему – величина.

### Запись

Запись — структурированный тип данных, позволяющие объединить данные разных типов. Записи являются неоднородными неупорядоченными структурами с прямым доступом к компонентам. Компоненты записи называют полями записи.

В данной программе используются записи для хранения информации о фильмах. Каждая запись состоит из полей, содержащих следующую информацию: адрес файла с изображением постера фильма, название фильма, сценарист, продюсер, год выпуска, жанр фильма, бюджет, выручка, количество просмотров в кинотеатре, длительность, краткое описание фильма, указатель на следующую запись. Все записи, за исключением указателя на следующую запись, строкового типа. Указатель на следующую запись списка имеет тип указателя на данную запись.

### Абстрактный тип данных «список»

Наиболее простой способ связать некоторое множество элементов - это организовать однонаправленный список. При такой организации элементы некоторого типа образуют цепочку. Для связывания элементов в списке используют систему указателей. В случае однонаправленного списка любой элемент списка имеет один указатель, который указывает на следующий элемент в списке или является пустым указателем, что означает конец списка. Списки являются чрезвычайно гибкой структурой.

Над динамическими списками определены три операции:

- поиск элемента в списке по выбранному полю;

- вставка элемента в список;

- удаление элемента из списка.

При разработке программы необходимо реализовать операции, не являющиеся базовыми операциями над списком, такие, как редактирование элемента в списке и сортировка списка по выбранному полю.

### Типизированные файлы

Типизированный файл - последовательность элементов одного типа и одной длины. В отличие от текстовых файлов, в типизированном файле информация хранится в двоичном представлении (в текстовом файле информация хранится в виде цепочки символов). Данный факт позволяет производить чтение и запись в файл более быстрым способом, т.к. отсутствует необходимость в преобразовании данных. Кроме того, в типизированном файле возможен прямой доступ к записям файла (в текстовом файле возможен только последовательный доступ).

Типизированный файл определяется следующим образом:

var

Variable: file of *<Тип\_файла>;*

Где тип либо предопределен, либо является пользовательским типом. В качестве типов не могут фигурировать динамические структуры, такие как динамические массивы, длинные строки или любые указатели.

### Текстовые файлы

Текстовый файл содержит последовательность символов (в основном печатных знаков, принадлежащих тому или иному набору символов). Эти символы обычно сгруппированы в строки. В современных системах строки разделяются разделителями строк, в прошлом же применялось хранение строк в виде записей постоянной или переменной длины. Иногда конец текстового файла также отмечается одним или более специальными знаками, известными как маркеры конца файла.

Текстовые файлы обеспечивают простой и удобный способ сохранения текстовых данных. Они обеспечивают простой механизм для чтения и записи численных данных, сохраненных как текст. Этот тип файлов объявляется с помощью служебного слова TextFile, например:

Var

Variable: TextFile;

Компоненты текстовых файлов не индексируются, поэтому произвольный доступ к ним невозможен.

Для доступа к записям применяются процедуры reset, append, rewrite, read, readln, write, writeln. Первым параметром в любой из перечисленных процедур должна стоять файловая переменная.

## Анализ аналогичных программных средств

В настоящие время существует несколько популярных программных средств содержащих информацию о фильмах. Проанализируем две из них.

Самым популярным приложением в русскоязычном сегменте является «КиноПоиск».

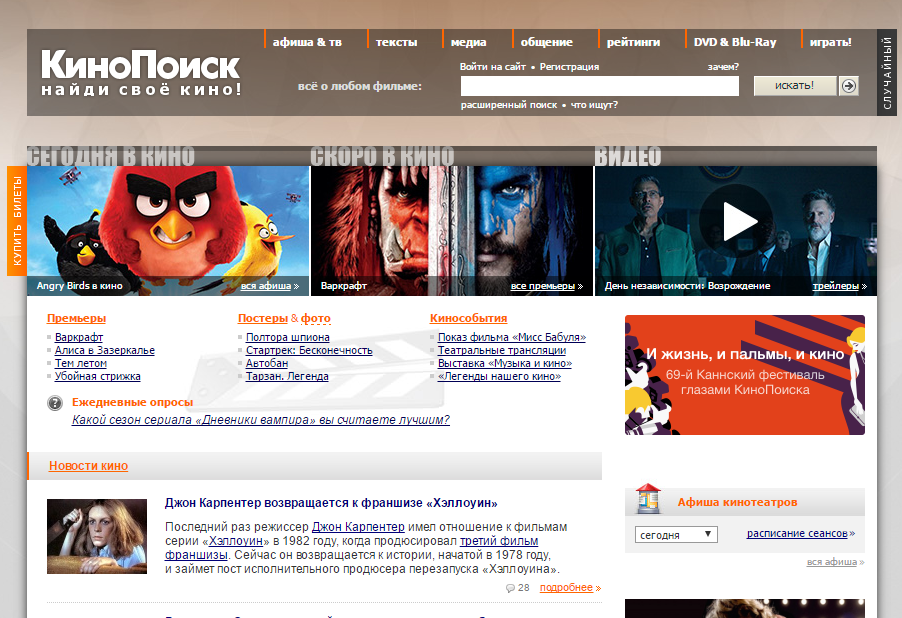


Рисунок 1.1 – Главное окно приложения

«КиноПоиск»

Программа содержит огромную подборку фильмов классифицированных по жанрам, годам производства, наградам и рейтингу.

В приложении содержится большое количество информации о фильмах. Представлены списки актеров и режиссеров. Присутствует расширенная информация о забавных моментах со съёмочной площадки, трейлеры к фильмам, постеры, информация о студии. В приложении можно прочитать рецензии зрителей и информацию о последних новинках киноиндустрии.

Однако в приложении отсутствует функции создания своей собственной подборки фильмов.

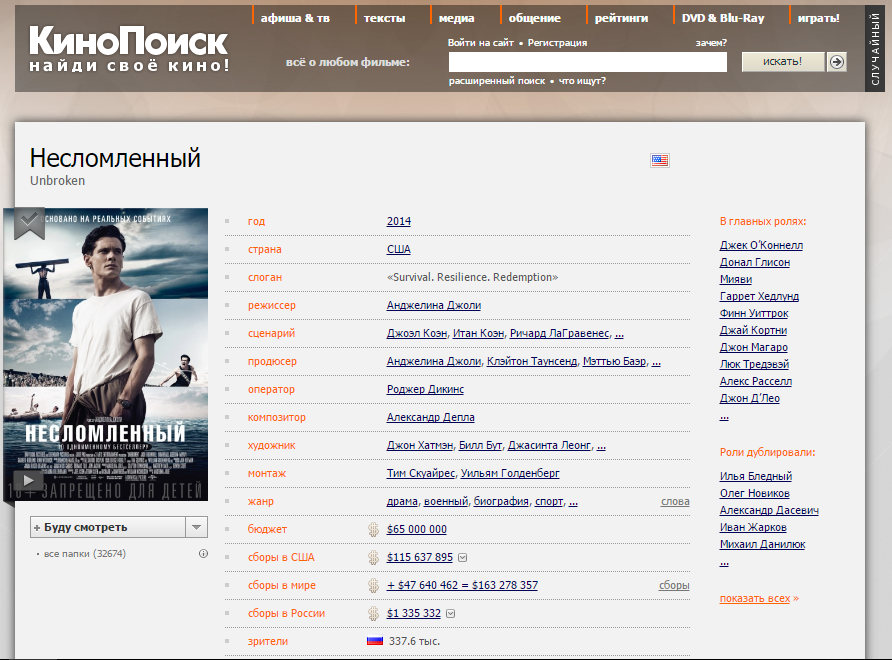


Рисунок 1.2 – Просмотр информации о фильме в программе

«КиноПоиск»

Рассмотрим американский аналог программного средства фильмотека.

В программе подробно описан каждый фильм с указанием всех актеров, сценаристов, жанра фильма, года выпуска, студий и возрастного ограничения. Имеется возможность просматривать фильмы в произвольном порядке и сортированных по году выпуска и сценаристам. Также имеется возможность прочитать информацию о последних новинках. Узнать дополнительную информацию об участниках фильм.

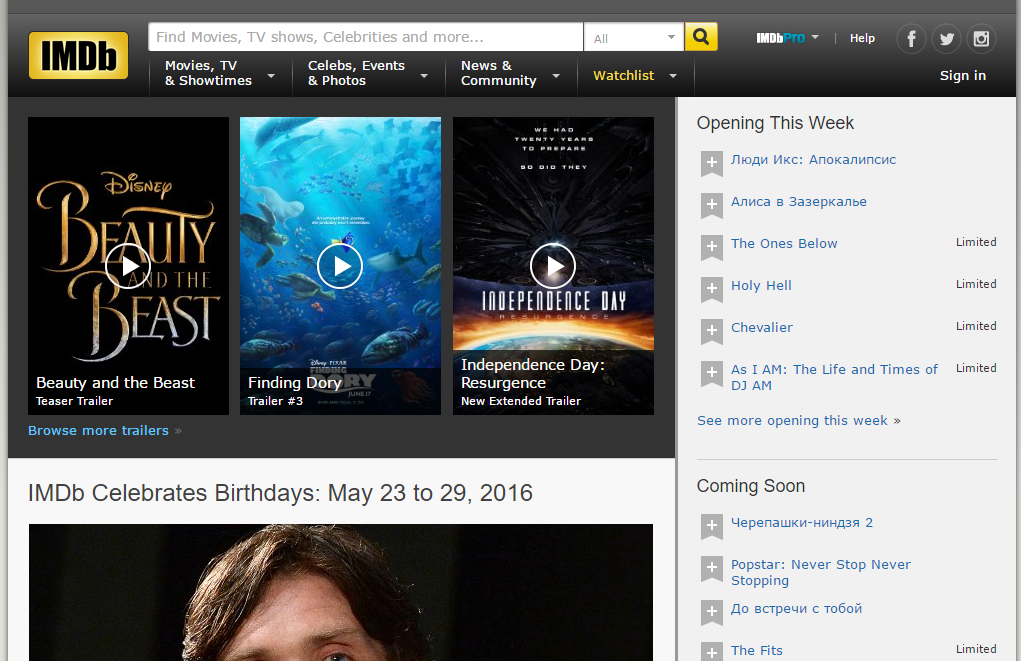
****

Рисунок 1.3 – Главное окно приложения

«КиноПоиск»

Однако неудобно, что информация не сгруппирована в одном месте и её сложно сразу заметить

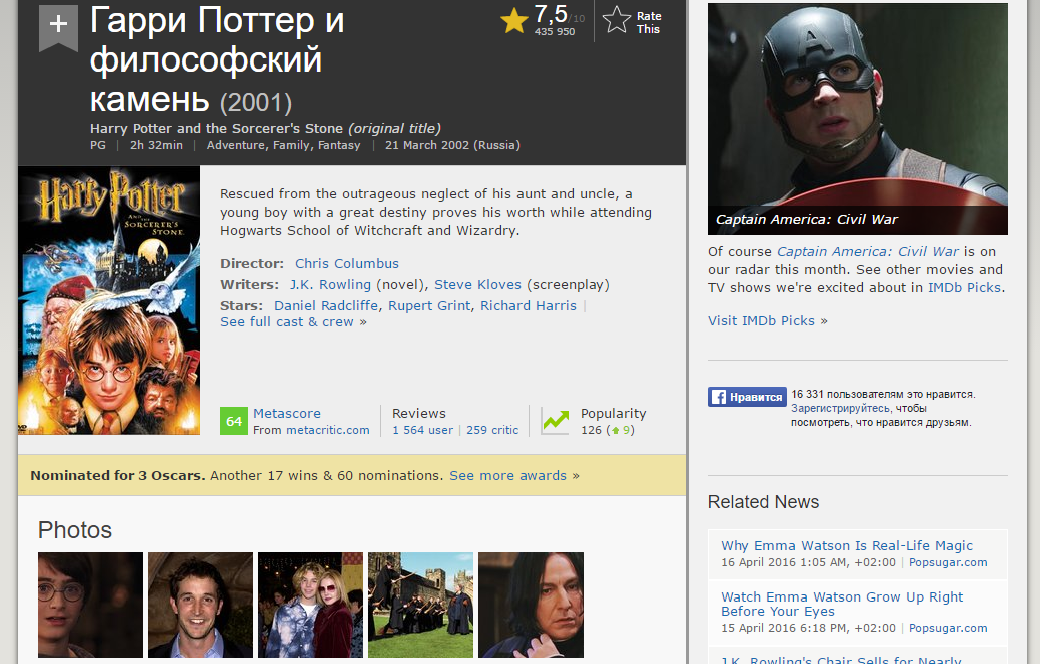
****

Рисунок 1.4 – Просмотр информации о фильме в программе «imdb»

# Постановка задачи

Целью данной работы является создание в среде Delphi приложения, позволяющего хранить информацию о фильмах. В качестве задач можно определить:

* Возможность добавления, удаления, редактирование списка с фильмами.
* предусмотреть возможность вывода информации о фильме на экран;
* организовать поиск фильмов по названию;
* предусмотреть возможность сортировки фильмов по жанру, стране производства и длительность;
* возможность экспорта списка фильмов в текстовый файл;
* возможность вывода информации о программе и разработчике;
* использовать динамические структуры данных, типизированные и текстовые файлы.

# Разработка программного средства

Программа – «Фильмотека» была разработана в среде Borland Delphi 7.

Delphi — интегрированная среда разработки программного обеспечения корпорации Borland. Delphi является средой RAD (rapid application development). По сути является наследником языка Pascal с объектно-ориентированными расширениями. Изначально язык был предназначен исключительно для разработки приложений Microsoft Windows, затем был реализован также для платформ Linux. Существенной чертой Delphi является компонентная модель разработки программных продуктов. Суть модели заключается в поддержке системой постоянно расширяемого набора объектных компонентов, из которых и строится программа.

Для хранения информации о фильмах в программе используется динамический однонаправленный список. Для реализации, был объявлен тип данных запись. Каждая запись состоит из полей, содержащих следующую информацию: адрес файла с изображением постера фильма, название фильма, сценарист, продюсер, год выпуска, жанр фильма, бюджет, выручка, количество просмотров в кинотеатре, длительность, краткое описание фильма, указатель на следующую запись. Все записи, за исключением указателя на следующую запись, строкового типа. Указатель на следующую запись списка имеет тип указателя на данную запись.

## Структура главного меню

В программе необходимо реализовать следующие процедуры: добавление, удаление и редактирование информации о фильме, сортировка по названию, жанру, стране производителю и длительности, поиск фильмов по названию, вывод информации о фильмах в текстовый файл, а также вывод информации о программе и её разработчике. Следовательно, структуру главного меню программы можно представить в виде схемы.

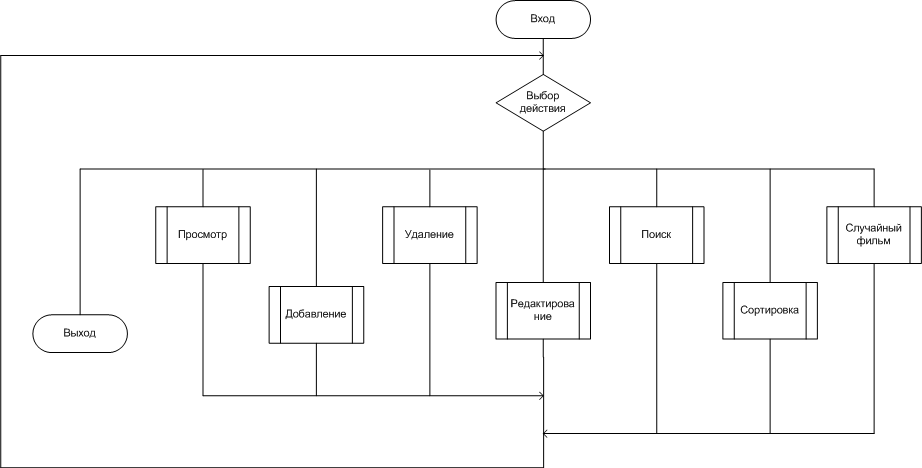


Рисунок 3.1 – Структура главного меню

## Структура программного средства

В процессе создания приложения о кинематографических произведения использовались следующие компоненты среды Delphi: Memo, Edit,SpinEdit, Button,BitBtn, Label, MainMenu, Image,ListView,panel.

Для удобного использования программы разобьем интерфейс на две области используя два компонента panel. Первый компонент panel содержит компоненты BitBtn. Каждый компонент BitBtn запускает выполнение одной из функций программы. Второй компонент panel содержит компонент ListView. Компонент ListView используется для отображения списка фильмов.

Компоненты Memo, Label, Edit, SpinEdit, Image используются для ввода и вывода информации.

## Структура типов

Таблица 1 – Структура типов

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы  данных | Рекомендуемый тип | Назначение | Комментарии |
| listFilm | Record  Name:string[30];  Year:string[10];  Genre:string[30];  pic:string[255];  Country:string[30];  Direct:string[50];  History:string[50];  Money:string[50]; MoneyUp:string[50]; Watch:string[50];  Time:string[50];  Disk:string[255];  next:TFilm;  end; | Тип для хранения информации о фильме Name, выпущенного в году Year, жанра Genre, с постером pic, страна производитель Country, режиссер фильма Direct, сценарист History, бюджет Money, кассовые сборы MoneyUp, количество просмотров Watch, длительность Time, описание Disk. |  |
| TFilm | ^listFilm | Указатель на listFilm |  |
| Func | function(First,second:ListFilm):boolean; | Тип-функция сравнивающая два элемента списка по выбранным критериям | Используется при сортировки списка фильмов |

## Структура данных

### 3.4.1 Структура данных основной программы3

Таблица 2 – Структура данных основной программы

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы  данных | Рекомендуемый тип | Назначение | Комментарии |
| FFileName | string | Путь к открытому файлу | В случае создания нового файла остается пустым до момента сохранения или загрузки |
| way | string | Директория программы |  |
| HeaderList | TFilm | Заглавный элемент однонаправленного списка | Используется для установки начала списка |
| ListOfFilm | TFilm | Элемент однонаправленного списка |  |
| Poster\_None | string | Путь к начальному постеру | Расположение картинки «отсутствия постера» |
| FLagChanges | boolean | Флаг изменения данных |  |

### 3.4.2 Структура данных алгоритма ChangeLV (HeaderList,ListOfFilm )

Таблица 3 – Структура данных алгоритма ChangeLV(HeaderList,ListOfFilm)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы  данных | Рекомендуемый тип | Назначение | Комментарии |
| HeaderList | TFilm | Заглавный элемент однонаправленного списка | Параметр - константа |
| ListOfFilm | TFilm | Элемент однонаправленного списка | Параметр - переменная |

### 3.4.3 Структура данных алгоритма SaveFilm (AlwaysAsk, changes, FFileName)

Таблица 4 – Структура данных алгоритма SaveFilm (AlwaysAsk, changes, FFilname)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы  данных | Рекомендуемый тип | Назначение | Комментарии |
| FileForSave | File of ListFilm | Файл для сохранения данных | Локальная переменная |
| FFileName | string | Название файла данных | Параметр - переменная |
| AlwaysAsk | Boolean | Флаг открытия диалога | Параметр - константа |
| Continue | Boolean | Флаг продолжения | Локальная переменная |
| changes | Boolean | Флаг изменения данных | Параметр - переменная |
| Data | ListFilm | Сохраняемые данные | Временно хранит данные |

### 3.4.4 Структура данных алгоритма ReadFile (FFileName, ListOfFilm, HeaderList)

Таблица 5 – Структура данных алгоритма ReadFile(FFileName, ListOfFilm)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы  данных | Рекомендуемый тип | Назначение | Комментарии |
| FFileName | string | Название файла с данными | Параметр - переменная |
| HeaderList | TFilm | Заглавный элемент однонаправленного списка | Параметр - константа |
| ListOfFilm | TFilm | Элемент однонаправленного списка | Параметр - переменная |
| FileForRead | File of ListFilm | Файл с данными | Локальная переменная |
| Data | ListFilm | Данные | Временно хранит считанные данные |
| temp | TFilm | Временный элемент списка | Локальная переменная |

### 3.4.5 Структура данных алгоритма Search(StrSearch, HeaderList, ListOfFilm)

Таблица 6 – Структура данных алгоритма Search ( StrSearch, HeaderList, ListOfFilm )

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы  данных | Рекомендуемый тип | Назначение | Комментарии |
| StrSearch | string | Строка поиска | Параметр - константа |
| HeaderList | TFilm | Заглавный элемент однонаправленного списка | Параметр - константа |
| ListOfFilm | TFilm | Элемент однонаправленного списка | Параметр - константа |
| ListItem | TlistItem | Элемент компонента listView | Локальная переменная |
| i | integer | Счетчик | Локальная переменная |

### 3.4.6 Структура данных алгоритма SortLV(HeaderList, ListOfFilm, SRav)

Таблица 7 – Структура данных алгоритма SortLV( HeaderList, ListOfFilm, SRav)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы  данных | Рекомендуемый тип | Назначение | Комментарии |
| HeaderList | TFilm | Заглавный элемент однонаправленного списка | Параметр - константа |
| ListOfFilm | TFilm | Элемент однонаправленного списка | Параметр - константа |
| SRav | Func | Функция сравнения элементов | Сравнивает два элемента по определенным требованиям |
| SortMass | array of ListFilm | Массив данных | Локальная переменная |
| search | boolean | Флаг | Локальная переменная |
| ArLength | integer | Размерность массива данных | Локальная переменная |
| ListItem | TlistItem | Элемент компонента listView | Локальная переменная |

### 3.4.7 Структура данных алгоритма showLV (HeaderList,ListOfFilm)

Таблица 8 – Структура данных алгоритма showLV(HeaderList,ListOfFilm)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы  данных | Рекомендуемый тип | Назначение | Комментарии |
| HeaderList | TFilm | Заглавный элемент однонаправленного списка | Параметр - константа |
| ListOfFilm | TFilm | Элемент однонаправленного списка | Параметр - константа |
| ListItem | TlistItem | Элемент компонента listView | Локальная переменная |

### 3.4.8 Структура данных алгоритма randomFilm ( HeaderList, ListOfFilm )

Таблица 9 – Структура данных алгоритма randomFilm(HeaderList,ListOfFilm)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы  данных | Рекомендуемый тип | Назначение | Комментарии |
| HeaderList | TFilm | Заглавный элемент однонаправленного списка | Параметр - константа |
| ListOfFilm | TFilm | Элемент однонаправленного списка | Параметр - константа |
| i | integer | Счетчик | Локальная переменная |

### 3.4.9 Структура данных алгоритма DeletLV(HeaderList, ListOfFilm)

Таблица 10 – Структура данных алгоритма DeletLV(HeaderList, ListOfFilm)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы  данных | Рекомендуемый тип | Назначение | Комментарии |
| HeaderList | TFilm | Заглавный элемент однонаправленного списка | Параметр - константа |
| ListOfFilm | TFilm | Элемент однонаправленного списка | Параметр - переменная |
| Temp | TFilm | Временные данные | Локальная переменная |

## Описание алгоритмов

Таблица 11 – Описание алгоритмов

|  |  |  |  |
| --- | --- | --- | --- |
| № | Наименование  алгоритма | Назначение  алгоритма | Формальные  параметры |
| 1 | Основной  алгоритм | Инициализация параметров, создание необходимых для работы форм: главной FMain, форму редактирования FChangeData, форму поиска FSearch, форму сортировки FSort |  |
| 2 | AddToLV  (Sender) | Производит добавление данных | Sender |
| 3 | ChangeLV  (HeaderList, ListOfFilm) | Изменяет данные списка ListOfFilm. HeaderList – заглавный элемент списка | HeaderList, ListOfFilm |
| 4 | DeletLV  (HeaderList, ListOfFilm) | Удаляет элемент из списка ListOfFilm. HeaderList – заглавный элемент списка | HeaderList, ListOfFilm |
| 5 | exportsToTXT (HeaderList, ListOfFilm) | Экспортирует название фильмов из списка ListOfFilm. HeaderList – заглавный элемент списка | HeaderList, ListOfFilm |
| 6 | randomFilm (HeaderList, ListOfFilm) | Выдает случайный фильм из списка ListOfFilm. HeaderList – заглавный элемент списка | HeaderList, ListOfFilm |
| 7 | ReadFile(FFileName, ListOfFilm, HeaderList ) | Считывает данные из файла FFileName и загружает в список ListOfFilm. HeaderList – заглавный элемент списка | FFileName  HeaderList, ListOfFilm |
| 8 | SaveFilm  (AlwaysAsk, changes, FFileName, ListOfFilm, HeaderList) | Сохраняет данные из списка ListOfFilm в файл FFileName.  Changes – флаг изменения данных  AlwaysAsk – флаг для выбора директории  HeaderList – заглавный элемент списка. |  |
| 9 | Search (StrSearch, HeaderList, ListOfFilm) | Ищет фильмы с названием StrSearch в списке ListOfFilm.  HeaderList – заглавный элемент списка. | StrSearch HeaderList, ListOfFilm |

Продолжение таблицы 10

|  |  |  |  |
| --- | --- | --- | --- |
| 10 | showLV  (HeaderList, ListOfFilm) | Отображает информацию о фильмах из списка ListOfFilm. HeaderList – заглавный элемент списка. | StrSearch HeaderList, ListOfFilm |
| 11 | sortLV(HeaderList, ListOfFilm, SRav) | Сортирует список ListOfFilm по полям SRav и отображает результат сортировки. HeaderList – заглавный элемент списка. | HeaderList, ListOfFilm, SRav |
| 12 | StartSort(TypeSort) | Запускает сортировку по полям TypeSort | TypeSort |

## Схема алгоритма решения задачи по ГОСТ 19.701-90

**Схема алгоритма**