Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Основы алгоритмизации и программирования (ОАиП)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту на тему

Программное средство «Домашняя библиотека»

БГУИР КП I–40 01 01 410 ПЗ

Выполнил

студент гр. 151004 Данилов Ф.А.

Проверил: Фадеева Е.П.

Минск 2022

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

УТВЕРЖДАЮ

Заведующий кафедрой ПОИТ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

Лапицкая Н.В. 2022г.

ЗАДАНИЕ

по курсовому проектированию

Студенту *Данилову Фёдору Андреевичу*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Тема работы *Программное средство «Домашняя библиотека» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_­­­­­­­­\_\_\_\_\_\_\_\_\_\_\_\_

2. Срок сдачи законченной работы *27.05.2022г.*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3. Исходные данные к работе *Среда программирования Delphi. Возможность ввода данных с клавиатуры или чтением из типизированного и нетипизированного файла. Предусмотрение контроля корректности ввода данных, а именно: текстовые и числовые поля. Предоставление информации о программе из текстового файла. Возможность сохранения данных библиотеки в типизированный файл записей и нетипизированный файл. Реализована динамическая структура данных: однонаправленный список. Реализация функций работы с библиотекой: добавление книг, изменение книг, удаление книг, сохранение библиотеки в файлы, открытие библиотеки из файлов, сортировка таблицы книг, поиск по таблице книг\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

4. Содержание расчетно-пояснительной записки (перечень вопросов, которые подлежат разработке)

*Введение*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*1 Анализ литературных источников\_*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*2 Постановка задачи\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*3 Разработка программного средства\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*4 Тестирование и проверка работоспособности программного средства\_\_\_\_*

*5 Руководство по установке и использованию программного средства\_\_\_\_\_*

*Заключение \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Список использованных источников\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

\_*Приложения* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

5. Перечень графического материала (с точным обозначением обязательных чертежей и графиков)

*Схема алгоритма в формате А1*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

6. Консультант по курсовой работе *Фадеева Е.П.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

7.Дата выдачи задания *16.02.2022г.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

8. Календарный график работы над проектом на весь период проектирования (с обозначением сроков выполнения и процентом от общего обьема работы):

*Раздел 1. Введение к 28.02.2022г. – 10 % готовности работы;\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Раздел 2 к 15.03.2022г. – 30% готовности работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Раздел 3 к 15.04.2022г. – 60% готовности работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Раздел 4 к 10.05.2022г. – 80% готовности работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Раздел 5.Заключение. Приложения к 20.05.2022г. – 90% готовности работы;*

*оформление пояснительной записки и графического материала к 27.05.2022г. – 100% готовности работы.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Защита курсового проекта с 27.05.2022г. по 01.06.2022г.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

РУКОВОДИТЕЛЬ *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Фадеева Е.П.*

*(подпись)*

Задание принял к исполнению *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Данилов Ф.А. 16.02.2022г.*

*(дата и подпись студента)*

**СОДЕРЖАНИЕ**

[Введение 7](#_Toc104454111)

[1 Анализ литературных источников 8](#_Toc104454112)

[1.1 Анализ существующих аналогов 8](#_Toc104454113)

[1.1.1 Веб-приложение libib 8](#_Toc104454114)

[1.1.2 Приложение Book Collector 9](#_Toc104454115)

[1.1.3 Приложение Evergreen 10](#_Toc104454116)

[1.2 Анализ методов и способов разработки 11](#_Toc104454117)

[1.2.1 Запись 11](#_Toc104454118)

[1.2.2 Массив 11](#_Toc104454119)

[1.2.3 Линейный однонаправленный список 12](#_Toc104454120)

[1.2.4 Файлы 12](#_Toc104454121)

[2 Постановка задачи 15](#_Toc104454122)

[2.1 Назначение разработки 15](#_Toc104454123)

[2.2 Перечень функциональных требований 15](#_Toc104454124)

[2.2.1 Отображение библиотеки в виде списка книг 15](#_Toc104454125)

[2.2.2 Добавление книг в библиотеку 15](#_Toc104454126)

[2.2.3 Изменение книг в библиотеке 16](#_Toc104454127)

[2.2.4 Удаление книг из библиотеки 16](#_Toc104454128)

[2.2.5 Экспорт библиотеки в файл 16](#_Toc104454129)

[2.2.6 Импорт библиотеки из файла 16](#_Toc104454130)

[2.2.7 Сортировка библиотеки 16](#_Toc104454131)

[2.2.8 Поиск по ключевым словам 16](#_Toc104454132)

[2.2.9 Окно помощи 16](#_Toc104454133)

[2.3 Входные и выходные параметры 17](#_Toc104454134)

[2.3.1 Входные параметры: 17](#_Toc104454135)

[2.3.2 Выходные параметры: 17](#_Toc104454136)

[2.4 Состав и параметры технических и программных средств 17](#_Toc104454137)

[3 Разработка программного средства 18](#_Toc104454138)

[3.1 Описание алгоритмов решения задачи 18](#_Toc104454139)

[3.2 Структура данных 22](#_Toc104454140)

[3.2.1 Структура типов программы 22](#_Toc104454141)

[3.2.2 Структура данных программы 23](#_Toc104454142)

[3.2.3 Структура данных алгоритма Add 24](#_Toc104454143)

[3.2.4 Структура данных алгоритма SetById 24](#_Toc104454144)

[3.2.5 Структура данных алгоритма RemoveById 24](#_Toc104454145)

[3.2.6 Структура данных алгоритма GetById 25](#_Toc104454146)

[3.2.7 Структура данных алгоритма LibraryExport 25](#_Toc104454147)

[3.2.8 Структура данных алгоритма LibraryImport 26](#_Toc104454148)

[3.2.9 Структура данных алгоритма SearchDown 26](#_Toc104454149)

[3.2.10 Структура данных алгоритма SearchUp 26](#_Toc104454150)

[3.2.11 Структура данных алгоритма Swap 27](#_Toc104454151)

[3.2.12 Структура данных алгоритма SortGridByCols 27](#_Toc104454152)

[3.2.13 Структура данных алгоритма HelpTextSet 27](#_Toc104454153)

[3.3 Схема алгоритмов решения задач по ГОСТ 19.701-90 28](#_Toc104454154)

[3.3.1 Схема алгоритма Add 28](#_Toc104454155)

[3.3.2 Схема алгоритма SetById 29](#_Toc104454156)

[3.3.3 Схема алгоритма RemoveById 30](#_Toc104454157)

[3.3.4 Схема алгоритма GetById 31](#_Toc104454158)

[3.3.5 Схема алгоритма LibraryExport 32](#_Toc104454159)

[3.3.6 Схема алгоритма LibraryImport 35](#_Toc104454160)

[3.3.7 Схема алгоритма SearchDown 39](#_Toc104454161)

[3.3.8 Схема алгоритма SearchUp 41](#_Toc104454162)

[3.3.9 Схема алгоритма Swap 43](#_Toc104454163)

[3.3.10 Схема алгоритма SortGridByCols 44](#_Toc104454164)

[3.3.11 Схема алгоритма HelpTextSet 45](#_Toc104454165)

[4 Тестирование и проверка работоспособности программного средства 46](#_Toc104454166)

[4.1 Запуск программного средства 46](#_Toc104454167)

[4.1.1 Тест 1 46](#_Toc104454168)

[4.2 Запуск окна добавления книг 46](#_Toc104454169)

[4.2.1 Тест 1 46](#_Toc104454170)

[4.3 Запуск окна добавления книг 47](#_Toc104454171)

[4.3.1 Тест 1 47](#_Toc104454172)

[4.4 Запуск окна изменения книг 48](#_Toc104454173)

[4.4.1 Тест 1 48](#_Toc104454174)

[4.5 Проверка ввода в числовое поле 48](#_Toc104454175)

[4.5.1 Тест 1 48](#_Toc104454176)

[4.6 Проверка открытия диалоговых окон 49](#_Toc104454177)

[4.6.1 Тест 1 49](#_Toc104454178)

[4.6.2 Тест 2 50](#_Toc104454179)

[4.6.3 Тест 3 50](#_Toc104454180)

[4.7 Запуск окна помощи 51](#_Toc104454181)

[4.7.1 Тест 1 51](#_Toc104454182)

[4.8 Поиск книги 52](#_Toc104454183)

[4.8.1 Тест 1 52](#_Toc104454184)

[4.8.2 Тест 2 52](#_Toc104454185)

[4.8.1 Тест 3 52](#_Toc104454186)

[4.9 Сортировка таблицы по столбцу 53](#_Toc104454187)

[4.9.1 Тест 1 53](#_Toc104454188)

[4.9.2 Тест 2 54](#_Toc104454189)

[4.9.3 Тест 3 54](#_Toc104454190)

[4.10 Сохранение библиотеки 55](#_Toc104454191)

[4.10.1 Тест 1 55](#_Toc104454192)

[4.11 Открытие библиотеки 56](#_Toc104454193)

[4.11.1 Тест 1 56](#_Toc104454194)

[5 Руководство по установке и использованию программного средства 58](#_Toc104454195)

[5.1 Установка 58](#_Toc104454196)

[5.2 Использование 61](#_Toc104454197)

[5.2.1 Добавление книги 61](#_Toc104454198)

[5.2.2 Изменение книги 61](#_Toc104454199)

[5.2.3 Сохранение библиотеки 62](#_Toc104454200)

[5.2.4 Поиск книг 63](#_Toc104454201)

[5.2.5 Сортировка таблицы 63](#_Toc104454202)

[Заключение 64](#_Toc104454203)

[Список использованной литературы 65](#_Toc104454204)

[Приложение А 66](#_Toc104454205)

[Приложение Б 82](#_Toc104454206)

[Приложение В 86](#_Toc104454207)

[Приложение Г 88](#_Toc104454208)

[Приложение Д 93](#_Toc104454209)

Введение

С бесконечным увеличением поглощаемой человеком информации становится очень сложно запоминать изученный ранее материал. Для этого люди записывают информацию о литературе, которую они изучают.

В случае с книгами это становится проблематично, так как чтобы записать в блокнот точную информацию книги недостаточно записать лишь название и автора. Существует много различных атрибутов, которые надо учитывать при выборе книги. Примером таких атрибутов являются издатель, издание, жанр, описание, вид обложки и пр. Для решения таких задач создаются специальные программные средства, «Домашние библиотеки». Данные программные средства позволяют удобно следить за списком прочитанных книг, используя списки книг на основе файлов.

Домашняя библиотека – упорядоченная коллекция разнородных электронных документов (в том числе книг, журналов), снабжённых средствами навигации и поиска.

В реализации проекта попутно решались следующие задачи:

* создание динамической структуры данных;
* работа с файлами (текстовыми, типизированными, нетипизированными);
* запись/чтение данных в/из файл(а);
* разработка пользовательского интерфейса для реализации перечисленных функций.

В качестве языка программирования был выбран язык Delphi, а средой разработки – Embarcadero Delphi 10, которые изучаются в рамках предмета «Основы алгоритмизации и программирования».

# Анализ литературных источников

## Анализ существующих аналогов

### Веб-приложение libib

Это программное средство целесообразно рассматривать как один из ведущих аналогов, поскольку оно является облачной библиотекой книг, музыки и фильмов, позволяющей пользователям получить доступ к библиотеке на любом устройстве при наличии доступа к сети Интернет.



Рисунок 1.1 – интерфейс веб-приложения libib[1]

Список выполняемых функций:

1. Хранение домашней библиотеки;
2. Возможность добавления, удаления материала;
3. Возможность вести коллекции;
4. Возможность сортировать библиотеку по различным аттрибутам.

Достоинства:

1. Возможность рассматривать в библиотеке не только книги, но и музыку, и фильмы;
2. Возможность менять тему приложения;
3. Получение доступа к библиотеки с любого устройства через сеть Интернет;
4. Возможность поиска книг через онлайн библиотеку.

Недостатки:

1. Отсутствие возможности пользоваться библиотекой без доступа к сети Интернет;
2. Наличие подписки для доступа к библиотеке;
3. Наличие предела каталога;
4. Отсутствие возможности добавления материала, отсутствующего в уже созданных базах данных.

### Приложение Book Collector

Это программное средство целесообразно рассматривать его как один из ведущих аналогов, поскольку оно является одним из лучших оффлайн приложений для учета книг.

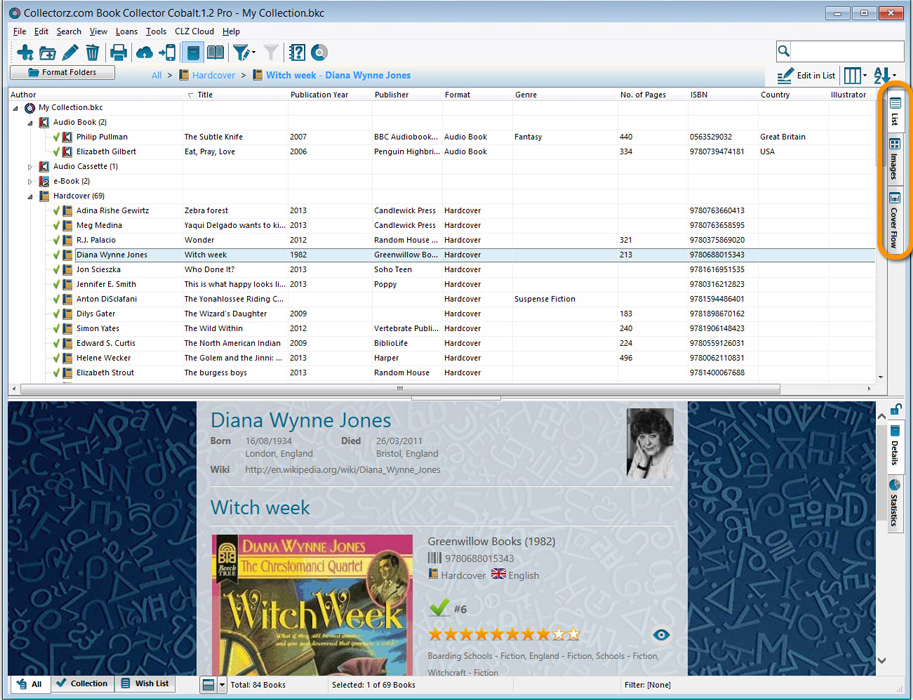


Рисунок 1.2 – интерфейс Book Collector[2]

Список выполняемых функций:

1. Хранение домашней библиотеки;
2. Возможность добавления, удаления книг;
3. Возможность вести коллекции;
4. Возможность сортировать библиотеки по различным параметрам.

Достоинства:

1. Наличие версий для всех популярных версий ОС Windows;
2. Наличие возможности сохранять библиотеку на компьютере;
3. Наличие возможности сохранять библиотеку в облачных хранилищах в сети Интернет;
4. Наличие автоматического поиска книг на компьютере для добавления в домашнюю библиотеку.

Недостатки:

1. Устаревший интерфейс;
2. Плохая совместимость с Windows 11;
3. Требуется подписка для использования приложения.

### Приложение Evergreen

Это программное средство целесообразно рассматривать как один из ведущих аналогов, поскольку оно занимает лидирующее положение среди бесплатных домашних библиотек.

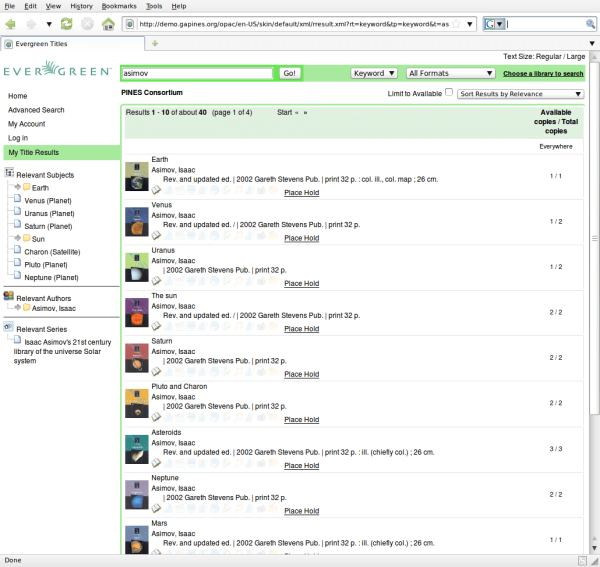


Рисунок 1.3 – интерфейс Evergreen[3]

Список выполняемых функций:

1. Хранение домашней библиотеки;
2. Возможность добавления, удаления книг;
3. Возможность вести коллекции;
4. Возможность сортировать библиотеки по различным параметрам.

Достоинства:

1. Приложение бесплатно;
2. Открытый исходный код.

Недостатки:

1. Отсутствие дальнейшей поддержки приложения;
2. Частые зависания приложения.

Исходя из превосходств и недостатков аналогов, очевидно, что приложение Book Collector является лучшим выбором, поэтому он будет являться ориентиром при разработке собственного программного средства.

## Анализ методов и способов разработки

Предполагается, что данное средство, помимо выполнения функций добавления и удаления книг в базу данных, будет выполнять ряд дополнительных функций, таких как поиск книг, сортировка по ряду критериев и т.д.

В данном программном средстве будут использоваться пять основных структур данных:

* Записи;
* Массивы;
* Линейные однонаправленные связные списки;
* Файлы.

### Запись

Запись – это структура данных, состоящая из фиксированного числа элементов, которые являются полями. В пределах записи имя поля должно быть уникальным. Описание записи происходит с помощью зарезервированного слова Record и дает возможность объединять данные разных типов вместе.

### Массив

Массивом называется упорядоченная совокупность однотипных элементов, у которых общее имя. Массивы в языке Delphi во многом схожи с аналогичными типами данных в других языках программирования. В состав элементов массива могут входить данные любых типов, в том числе и структурированные. Любой элемент массива определяется именем массива и индексом (индекс может быть не один, тогда данный массив будет многомерным). В качестве индексных типов в языке Delphi используются перенумерованные типы. Обычно используется тип Integer.

### Линейный однонаправленный список

Линейный однонаправленный список – динамическая структура данных, состоящая из элементов одного типа, связанных между собой последовательно посредством указателей. Каждый элемент списка имеет указатель на следующий. Последний элемент списка указывает на Nil. Элемент, на который нет указателя является первым (головным) элементом списка. Ссылка в каждом узле указывает на следующий узел в списке. В односвязном списке можно передвигаться только в сторону конца списка. Узнать адрес предыдущего элемента, опираясь на содержимое текущего узла, невозможно.

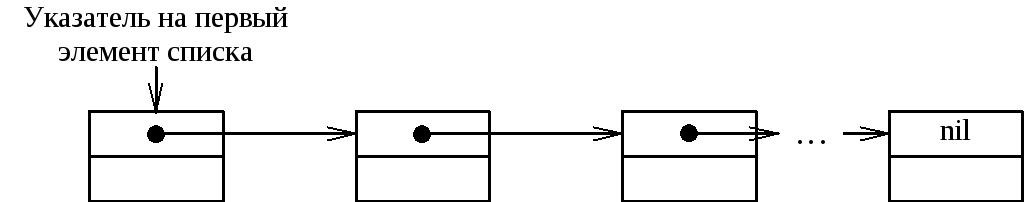


Рисунок 1.4 – Иллюстрация структуры линейного однонаправленного связного списка

### Файлы

Файловый тип — это произвольная последовательность элементов, длина которой заранее не определена, а конкретизируется в процессе выполнения программы. Это определение логического файла, т.е. того, который используется в программе (файл с точки зрения программиста). Физический файл (набор данных) — это именованная область памяти на внешнем носителе, в которой хранится некоторая информация (файл с точки зрения пользователя).

B Delphi существует три типа файлов:

* типизированные файлы;
* текстовые файлы;
* нетипизированные файлы.

Типизированные файлы связываются с файловыми переменными, объявленными как file of <Тип>. Файл считается состоящим из элементов, каждый из которых имеет тип «Тип>.

Нетипизированные файлы могут быть связаны только с теми файловыми переменными, которые были объявлены как file. Файл считается состоящим из элементов, размер которых определяется при открытии файла.

Текстовый файл представляет собой последовательность символов, однако он не эквивалентен файлу типа file of Char. Файл типа file of Char — это единая последовательность символов. Текстовые файлы связываются с файловыми переменными, принадлежащими стандартному типу TextFile. Особенность текстовых файлов состоит в том, что содержащиеся в них символы разбиваются на строки. Строки могут быть различной длины (пустые в том числе). В конце каждой строки помещается специальный управляющий символ: возврат каретки (#13 или ^М - международное обозначение CR) и перехода новую строку (#10 или ^J - международное обозначение LF). С наличием этого маркера связана логическая функция Eoln (End of line).

Для доступа к отдельным элементам файлов в Delphi существуют специальные стандартные процедуры и функции. Их называют процедурами и функциями ввода-вывода. Обращение к ним осуществляется обычным образом.

Процедура AssignFile(F, Name) связывает файловую переменную с внешним файлом на диске. Здесь F - имя файловой переменной любого типа; Name - выражение строкового типа. Назначение процедуры следующее: организация связи между конкретным физическим файлом на внешнем устройстве (конкретным набором данных) и файловой переменной (логическим файлом) F.

Процедура Reset(F) открывает существующий файл F в режиме чтения. Фактически при этом открывается внешний файл с именем, присвоенным переменной F процедурой AssignFile. Если файл с данным именем не существует, возникает исключение. Указатель файла устанавливается на первую позицию файла. Если файл был предварительно открыт, то он закрывается и повторно открывается. Процедура Reset(F) может быть применена к файлу любое количество раз. При выполнении данной процедуры содержимое файла не изменяется. Для текстовых файлов - в режиме только для чтения (read-only).

Процедура Rewrite(F) создает новый файл и открывает его. Если файл уже открыт, то закрывает и снова открывает его. Указатель файла устанавливается на начало файла (файл создается пустым). Для текстовых файлов - в режиме только для записи (write-only).

Процедура Append(F) предопределена только для текстовых файлов. Она открывает существующий файл для добавления. Если файл уже открыт, то закрывает его, а затем вновь открывает. Если файл с данным именем не существует, возникает исключение. В данном случае указатель файла устанавливается на конец файла. Файл открывается в режиме только для записи (write-only).

Процедура CloseFile(F) закрывает файл. В данном случае F - файловая переменная, открытая с помощью Reset(F), Rewrite(F) или Append(F).

Функция Eof(F) проверяет, является ли текущей позицией конец файла, и возвращает результат логического типа. В данном случае F - файловая переменная, открытая с помощью Reset(F) или Rewrite(F).

Функция Eoln(F) проверяет, является ли текущей позицией конец строки, и возвращает результат логического типа. В данном случае F - файловая переменная, открытая с помощью Reset(F), Rewrite(F) или Append(F).

Для файлов с типом предопределены процедуры считывания компонентов файла в переменные и записи в файл компонентов из переменных Read (F, V1, V2, …, Vn) и Write (F, V1, V2, …, Vn).

Для файлов без типа есть аналоги процедур Read и Write. Это процедуры BlockRead (F, Buf, Count [, Done]) и BlockWrite (F, Buf, Count [, Done]).

Также есть функции и процедуры, перемещающие указатель файла в указанную позицию, возвращающие текущее положение указателя файла в байтах, возвращающие размер файла в компонентах и др.

# Постановка задачи

## Назначение разработки

С бесконечным увеличением поглощаемой человеком информации становится очень сложно запоминать изученный ранее материал. Для этого люди записывают информацию о материале, который они изучают.

В случае с книгами это становится проблематично, так как чтобы записать в блокнот точную информацию книги недостаточно записать лишь название и автора. Существует много различных атрибутов, которые надо учитывать при выборе книги. Примером таких атрибутов являются издатель, издание, жанр, описание, вид обложки и пр. Для решения таких задач создаются специальные программные средства, «Домашние библиотеки». Данные программные средства позволяют удобно следить за списком прочитанных книг, используя списки книг на основе файлов.

## Перечень функциональных требований

После анализа аналогов был составлен следующий список выполняемых функций:

### Отображение библиотеки в виде списка книг

Библиотека будет отображаться в виде таблицы, в которой строками являются книги, а столбцами – атрибуты книг. В программном средстве используется следующий список атрибутов:

* Название
* Автор книги
* Жанр
* Описание
* Издатель
* Дата выхода
* Количество страниц
* Обложка
* Язык
* Издание
* Вид переплета

### Добавление книг в библиотеку

Для добавления книг в библиотеку открывается окно, в котором находятся поля для ввода атрибутов.

### Изменение книг в библиотеке

Для изменения книг в библиотеке открывается окно, в котором находятся поля для изменения атрибутов.

### Удаление книг из библиотеки

Для удаления книг из библиотеки надо выбрать книгу в таблице и нажать на кнопку удалить книгу.

### Экспорт библиотеки в файл

Для экспорта книг из библиотеки надо нажать на кнопку. При нажатии на кнопку открывается окно сохранения файлов Windows.

### Импорт библиотеки из файла

Для импорта книг из библиотеки надо нажать на кнопку. При нажатии на кнопку открывается окно открытия файлов Windows.

### Сортировка библиотеки

Для сортировки библиотеки надо 2 раза нажать по заголовку столбца, который требуется отсортировать. Сортировка выполняется по строковому типу.

### Поиск по ключевым словам

Для поиска книг по ключевым словам надо в поле поиска ввести ключевое слово и выбрать атрибут.

### Окно помощи

Окно помощи представляет собой окно, в котором описывается программа и как ей пользоваться. Информация из окна помощи хранится в текстовом файле.

## Входные и выходные параметры

### Входные параметры:

Файл библиотеки для импорта, книги с атрибутами

### Выходные параметры:

Файл библиотеки для экспорта, отображения библиотеки в таблице

## Состав и параметры технических и программных средств

Программное средство «Домашняя библиотека» должно функционировать на персональных компьютерах со следующими характеристиками:

* Процессор Ryzen 7 4700U 4.1 GHz или лучше;
* оперативная память 16 GB 2600 MHz DDR4 или лучше;
* накопитель SSD M.2 объемом 512 GB или больше.

Программное средство «Домашняя библиотека» должно функционировать в окружении операционной системы Windows 11 Pro 21H2.

# Разработка программного средства

## Описание алгоритмов решения задачи

Таблица 3.1 – Описание алгоритмов решения задачи

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| №  п.п. | Наименование алгоритма | Назначение алгоритма | Формальные параметры | Предлагае-мый тип реализации |
| 1 | Инициализация основной рабочей формы | Производит создание и загрузку фильмотеки.  Добавляет, удаляет и редактирует книги.  Предоставляет список книг.  Производит сортировку, и быстрый поиск книг.  Сохраняет/открывает библиотеку.  Использует алгоритмы:  Add, SetById, RemoveById,  GetById,  LibraryExport,  LibraryImport,  SearchDown, SearchUp,  Swap, SortGridByCols,  HelpTextSet |  |  |
| 2 | Add  (List, Data) | Добавляет в список книг запись, содержащую поля  Name, Author, Description, Genre, Publisher, Language, Edition, Binding, Date, Pages, Cover | List – получает от фактического параметра адрес.  Data – получает от фактического параметра значение | Процедура. |

Продолжение таблицы 3.1 – Описание алгоритмов решения задачи

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | SetById  (List, Id, Data) | Изменяет значение элемента списка List под номером Id на запись Data, содержащую поля Name, Author, Description, Genre, Publisher, Language, Edition, Binding, Date, Pages, Cover | List – получает от фактического параметра адрес.  Id – получает от фактического параметра значение.  Data – получает от фактического параметра значение | Процедура |
| 4 | RemoveById  (List, Id) | Удаляет элемент списка List под номером Id | List – получает от фактического параметра адрес.  Id – получает от фактического параметра значение | Процедура |
| 5 | GetById(List, Id, Result) | Возвращает элемент Result под номером Id из списка List | List – получает от фактического параметра адрес.  Id – получает от фактического параметра значение.  Result – получает от фактического параметра адрес | Функция. Result – возвращаемый параметр |

Продолжение таблицы 3.1 – Описание алгоритмов решения задачи

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 6 | LibraryExport  (List, FileName) | Сохраняет список List на память компьютера в папку FileName | List – получает от фактического параметра адрес.  FileName – получает от фактического параметра значение | Процедура |
| 7 | LibraryImport  (List, FileName) | Осуществляет поиск места в динамическом списке для вставки нового элемента типа запись | List – получает от фактического параметра адрес.  FileName – получает от фактического параметра значение | Процедура |
| 8 | SearchDown  (Grid, Text) | Производит поиск Text по таблице Grid вниз | Grid – получает от фактического параметра адрес.  Text – получает от фактического параметра значение | Процедура |
| 9 | SearchUp  (Grid, Text) | Производит поиск Text по таблице Grid вверх | Grid – получает от фактического параметра адрес.  Text – получает от фактического параметра значение | Процедура |

Продолжение таблицы 3.1 – Описание алгоритмов решения задачи

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 10 | Swap(Grid, Row1, Row2) | Меняет местами строки под номерами Row1 и Row2 в таблице Grid | Grid – получает от фактического параметра адрес.  Row1 – получает от фактического параметра значение.  Row2 – получает от фактического параметра значение | Процедура |
| 11 | SortGridByCols (Grid, Col, Flag) | Сортирует таблицу Grid по столбцу Col в зависимости от значения Flag.  Flag – true сортирует  таблицу по возрастанию, иначе – по убыванию | Grid – получает от фактического параметра адрес.  Col – получает от фактического параметра значение.  Flag – получает от фактического параметра значение | Процедура |
| 12 | HelpTextSet  (MemoHelp) | Запись текста помощи из файла в MemoHelp | MemoHelp – получает от фактического параметра адрес | Процедура |

## Структура данных

### Структура типов программы

Таблица 3.2 – Структура типов программы

|  |  |  |
| --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение |
| TItem | Record  Title: String  Author: String  Description: String  Genre: String  Publisher: String  Language: String  Edition: String  Binding: String  Date: TDate  Pages: Integer  Cover: String  End; | Информация о книге в списке:  Title – название;  Description – описание;  Genre – жанр;  Publisher – издатель;  Language – язык;  Edition – издание;  Binding – переплёт;  Date – дата выхода;  Pages – количество страниц;  Cover – путь к обложке |
| TItemExport | Record  Title: String[255]  Author: String[255]  Genre: String[255]  Publisher: String[255]  Language: String[255]  Edition: String[255]  Binding: String[255]  Date: String[255]  Pages: Integer  End; | Информация о книге в файле:  Title – название;  Genre – жанр;  Publisher – издатель;  Language – язык;  Edition – издание;  Binding – переплёт;  Date – дата выхода;  Pages – количество страниц |
| TNode | Record  Data: TItem;  Next: PNode;  End; | Тип элемента, предназначенный для построения линейного однонаправленного списка, содержащего библиотеку.  Data – элемент списка: фильм;  Next – указатель на следующее звено списка |
| PNode | ^TNode | Типизированный указатель на элемент линейного однонаправленного списка |

Продолжение таблицы 3.2 – Структура типов программы

|  |  |  |
| --- | --- | --- |
| TList | Record  Head: PNode;  Length: Integer;  End; | Запись линейного однонаправленного списка, содержащая указатель на первый элемент и длину списка |

### Структура данных программы

Таблица 3.3 – Структура данных программы

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Комментарии |
| Form\_Main | TFormMain | Обеспечение графического интерфейса во время основной работы с программной | Главная форма программы |
| Form\_Add | TFormAdd | Обеспечение графического интерфейса во время добавления/изменения книги | Вспомогательная форма программы |
| Form\_Help | TFormHelp | Обеспечение графического интерфейса окна помощи | Вспомогательная форма программы |
| List | TList | Указатель на линейный однонаправленный список |  |
| SearchI | Integer | Столбец последнего найденного элемента |  |
| SearchJ | Integer | Строка последнего найденного элемента |  |

### Структура данных алгоритма Add

Таблица 3.4 – Структура данных алгоритма Add(List, Data)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| List | TList | Указатель на линейный однонаправленный список | Формальный |
| Data | TItem | Количество элементов в списке | Формальный |
| NodeNew | PNode | Элемент списка | Локальный |
| Node | PNode | Элемент списка | Локальный |

### Структура данных алгоритма SetById

Таблица 3.5 – Структура данных алгоритма SetById(List, Id, Data)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| List | TList | Указатель на линейный однонаправленный список | Формальный |
| Data | TItem | Количество элементов в списке | Формальный |
| Id | Integer | Номер элемента списка | Формальный |
| Node | PNode | Элемент списка | Локальный |
| I | Integer | Счетчик цикла | Локальный |

### Структура данных алгоритма RemoveById

Таблица 3.6 – Структура данных алгоритма RemoveById(List, Id)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| List | TList | Указатель на линейный однонаправленный список | Формальный |
| Id | Integer | Номер элемента списка | Формальный |
| Node1, Node2 | PNode | Элемент списка | Локальный |
| I | Integer | Счетчик цикла | Локальный |

### Структура данных алгоритма GetById

Таблица 3.7 – Структура данных алгоритма GetById(List, Id, Result)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| List | TList | Указатель на линейный однонаправленный список | Формальный |
| Id | Integer | Номер элемента списка | Формальный |
| Result | TItem | Данные элемента списка | Формальный |
| Node | PNode | Элемент списка | Локальный |
| i | Integer | Счетчик цикла | Локальный |

### Структура данных алгоритма LibraryExport

Таблица 3.8 – Структура данных алгоритма LibraryExport(List, FileName)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| List | TList | Линейный однонаправленный список | Формальный |
| FileName | String | Путь к файлу | Формальный |
| FileLibrary | File of TItemExport | Файл библиотеки | Локальный |
| FileDescriptions | File | Файл описаний книг | Локальный |
| ItemExport | TItemExport | Запись для сохранения в файл | Локальный |
| Item | TItem | Запись книги | Локальный |
| SavedCover | String | Путь к обложке книги по умолчанию | Локальный |
| i | Integer | Счетчик цикла | Локальный |
| WriteLength | Integer | Длина записи строки в файл описаний | Локальный |

### Структура данных алгоритма LibraryImport

Таблица 3.9 – Структура данных алгоритма LibraryImport(List, FileName)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| List | TList | Линейный однонаправленный список | Формальный |
| FileName | String | Путь к файлу | Формальный |
| FileLibrary | File of TItemExport | Файл библиотеки | Локальный |
| FileDescriptions | File | Файл описаний книг | Локальный |
| ItemExport | TItemExport | Запись для сохранения в файл | Локальный |
| Item | TItem | Запись книги | Локальный |
| i | Integer | Счетчик цикла | Локальный |
| Offset | Integer | Сдвиг чтения файла | Локальный |
| Length | Integer | Длина строки | Локальный |
| Description | String | Строка описания | Локальный |

### Структура данных алгоритма SearchDown

Таблица 3.11 – Структура данных алгоритма SearchDown(Grid, Text)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| Grid | TStringGrid | Таблица книг | Формальный |
| Text | String | Текст для поиска | Формальный |
| i | Integer | Счетчик цикла | Локальный |
| j | Integer | Счетчик цикла | Локальный |
| NotFound | Boolean | Состояние поиска | Локальный |

### Структура данных алгоритма SearchUp

Таблица 3.12 – Структура данных алгоритма SearchUp(Grid, Text)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| Grid | TStringGrid | Таблица книг | Формальный |
| Text | String | Текст для поиска | Формальный |
| i | Integer | Счетчик цикла | Локальный |
| j | Integer | Счетчик цикла | Локальный |
| NotFound | Boolean | Состояние поиска | Локальный |

### Структура данных алгоритма Swap

Таблица 3.13 – Структура данных алгоритма Swap(Grid, Row1, Row2)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| Grid | TStringGrid | Таблица книг | Формальный |
| Row1 | Integer | Индекс строки | Формальный |
| Row2 | Integer | Индекс строки | Формальный |
| i | Integer | Счетчик цикла | Локальный |
| Temp | String | Переменная строка для перестановки | Локальный |

### Структура данных алгоритма SortGridByCols

Таблица 3.14 – Структура данных алгоритма

SortGridByCols(Grid, Col, Flag)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| Grid | TStringGrid | Таблица книг | Формальный |
| Col | Integer | Номер столбца таблицы | Формальный |
| Flag | Boolean | Переменная, влияющая на результат сравнения.  Flag = true – сортировка по возрастания, иначе – по убыванию | Формальный |
| i | Integer | Счетчик цикла | Локальный |
| j | Integer | Счетчик цикла | Локальный |
| MaxInd | Integer | Индекс максимального элемента столбца | Локальный |

### Структура данных алгоритма HelpTextSet

Таблица 3.15 – Структура данных алгоритма HelpTextSet(MemoHelp)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| MemoHelp | TMemo | Текстовое поле | Формальный |
| HelpFile | TextFile | Текстовый файл помощи | Локальный |
| HelpText | String | Строка из файла | Локальный |

## Схема алгоритмов решения задач по ГОСТ 19.701-90

### Схема алгоритма Add

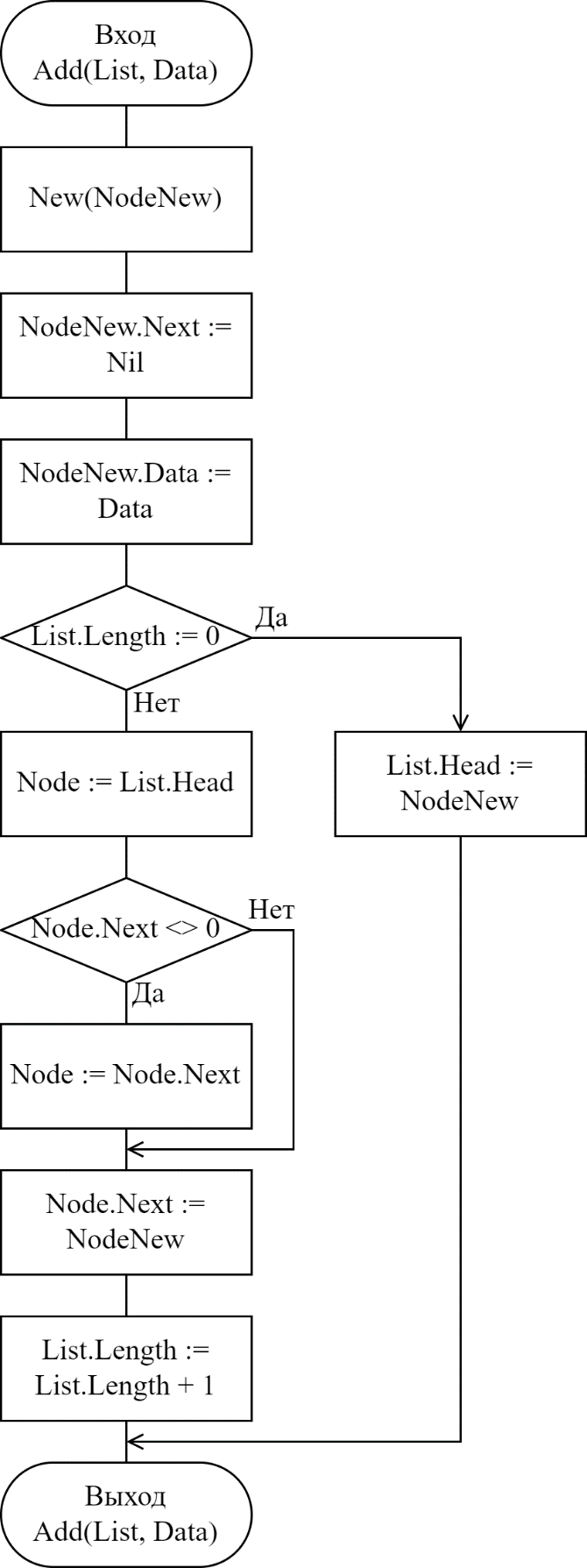


Рисунок 3.1 – Схема алгоритма Add

### Схема алгоритма SetById

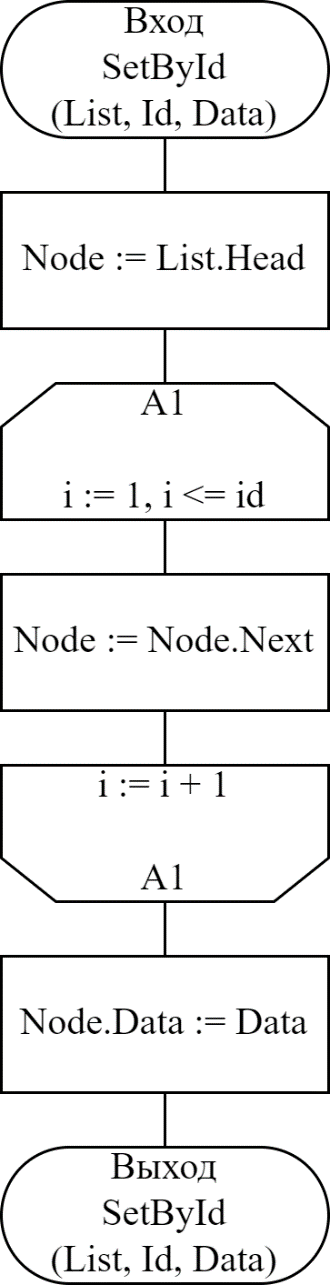


Рисунок 3.2 – Схема алгоритма SetById

### Схема алгоритма RemoveById

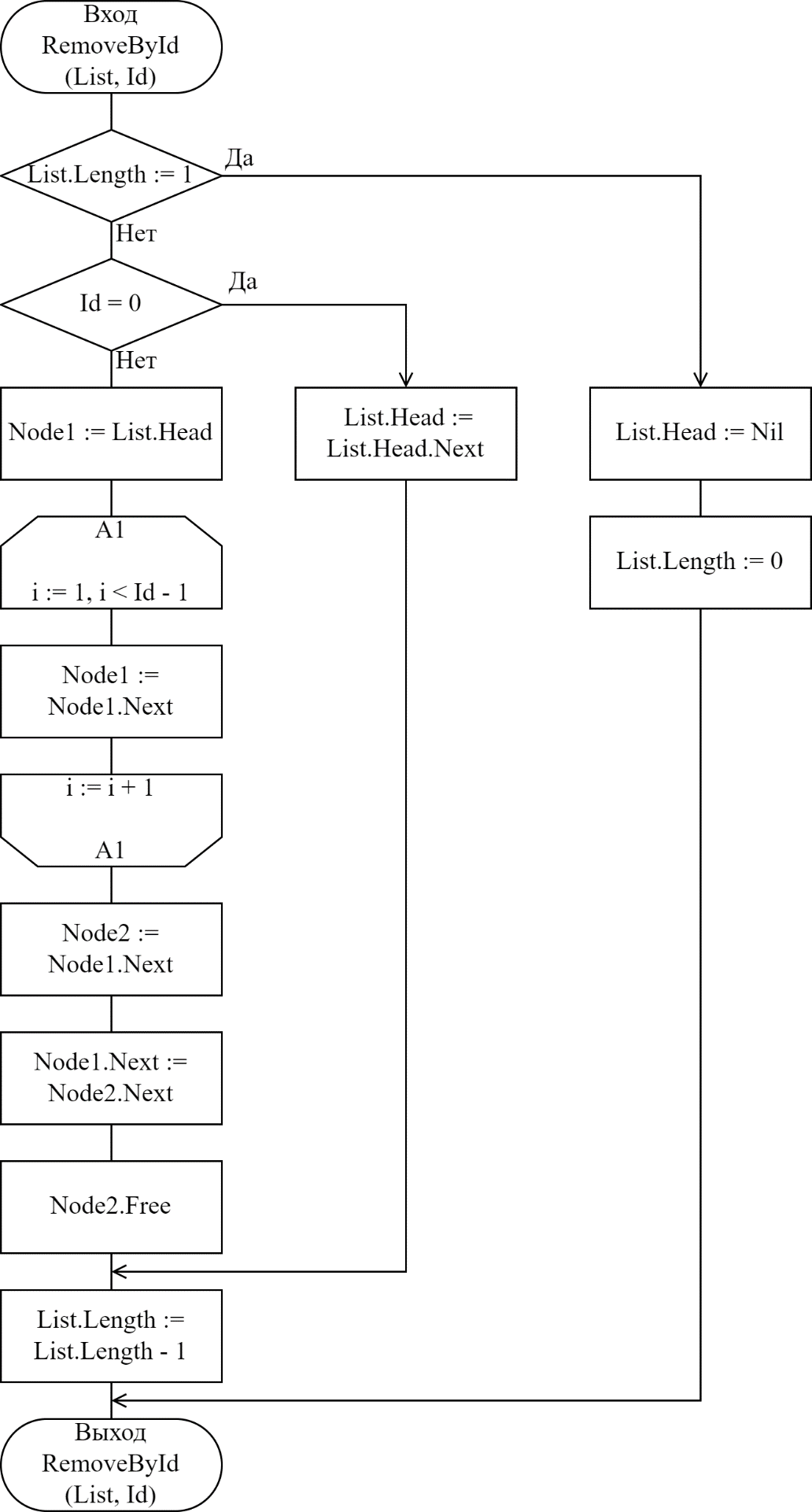


Рисунок 3.3 – Схема алгоритма RemoveById

### Схема алгоритма GetById

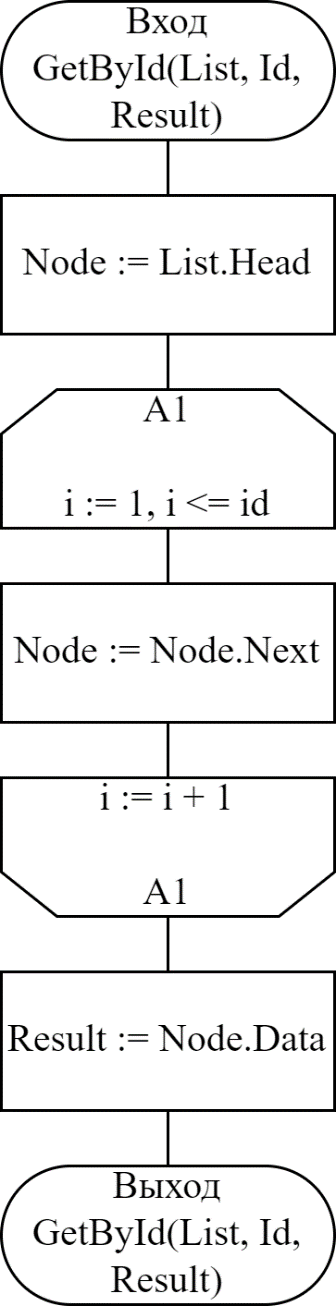


Рисунок 3.4 – Схема алгоритма GetById

### Схема алгоритма LibraryExport

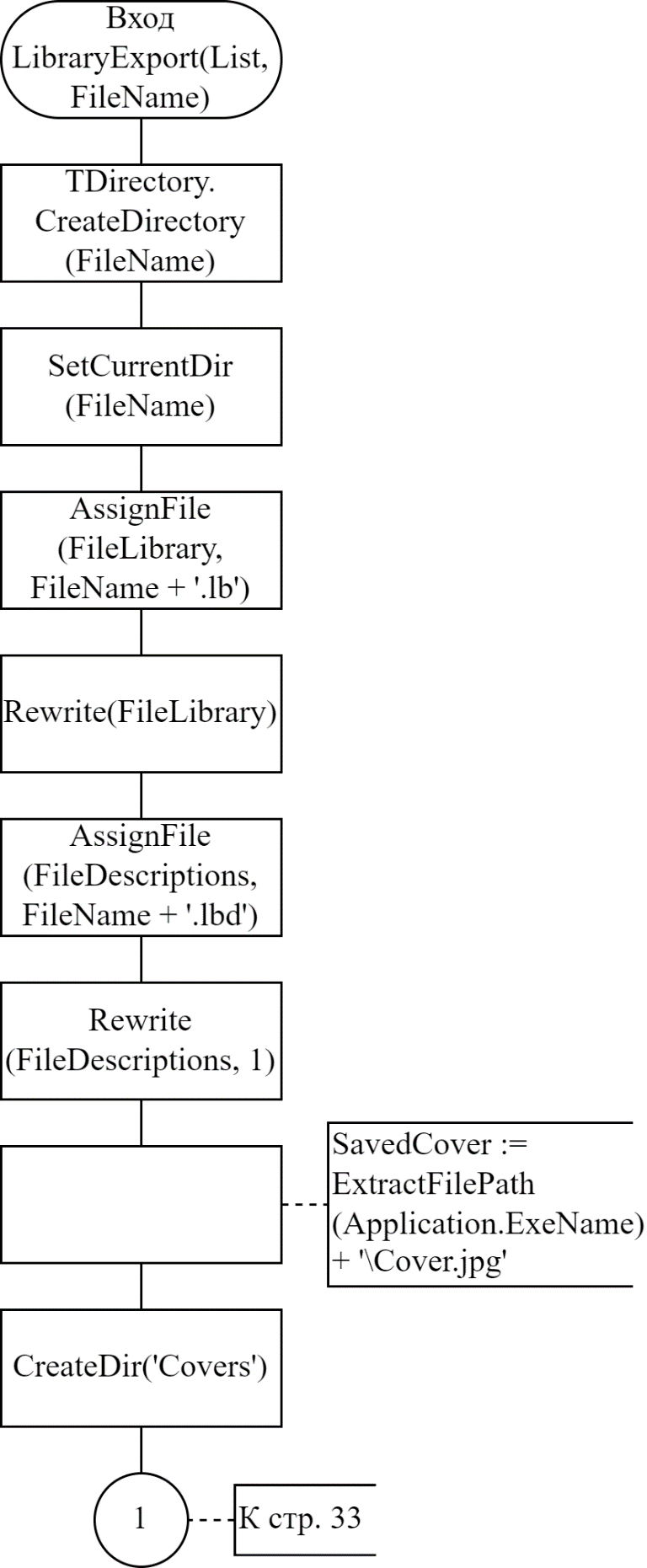


Рисунок 3.5 – Схема алгоритма LibraryExport (часть 1)

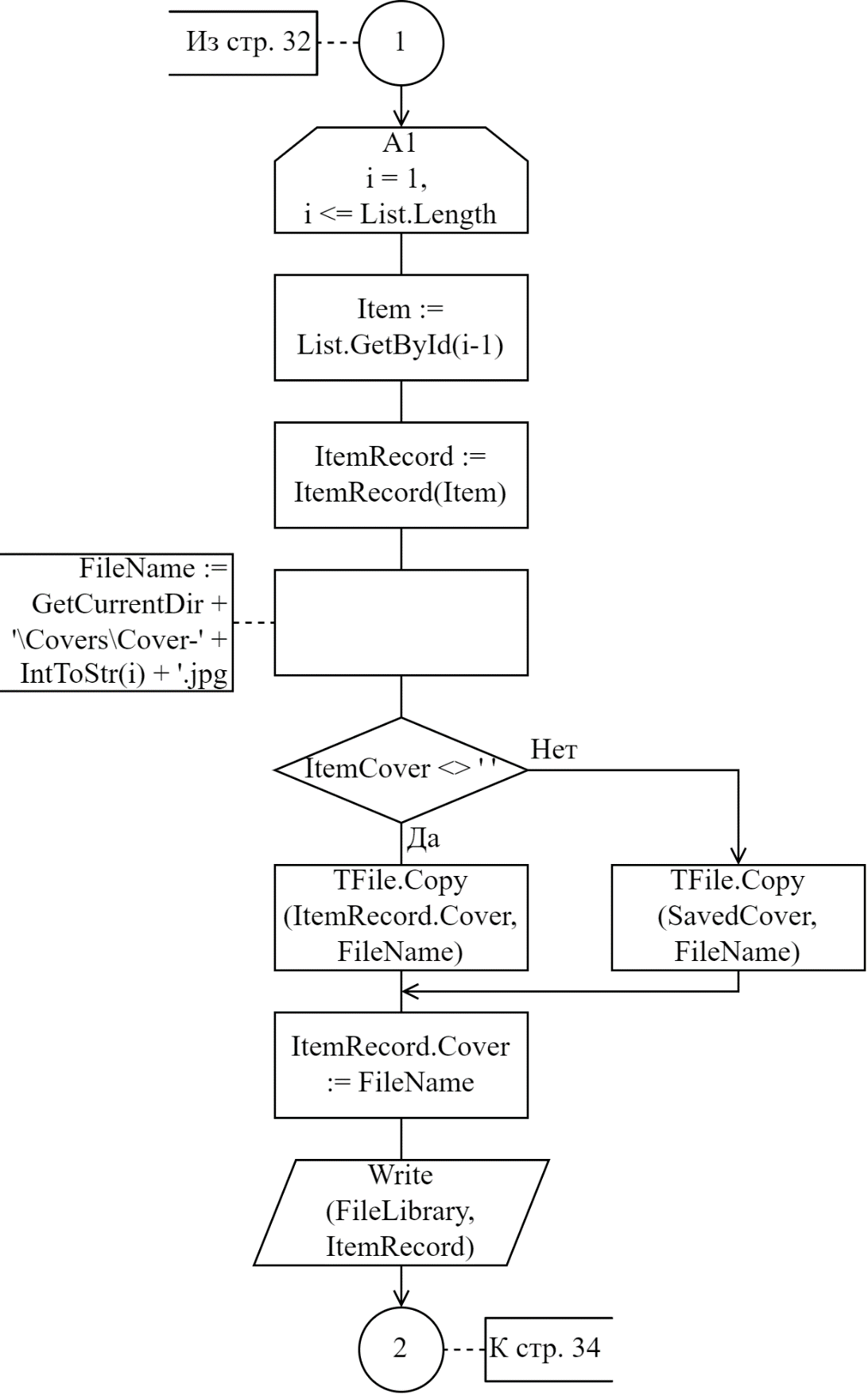
**

Рисунок 3.6 – Схема алгоритма LibraryExport (часть 2)

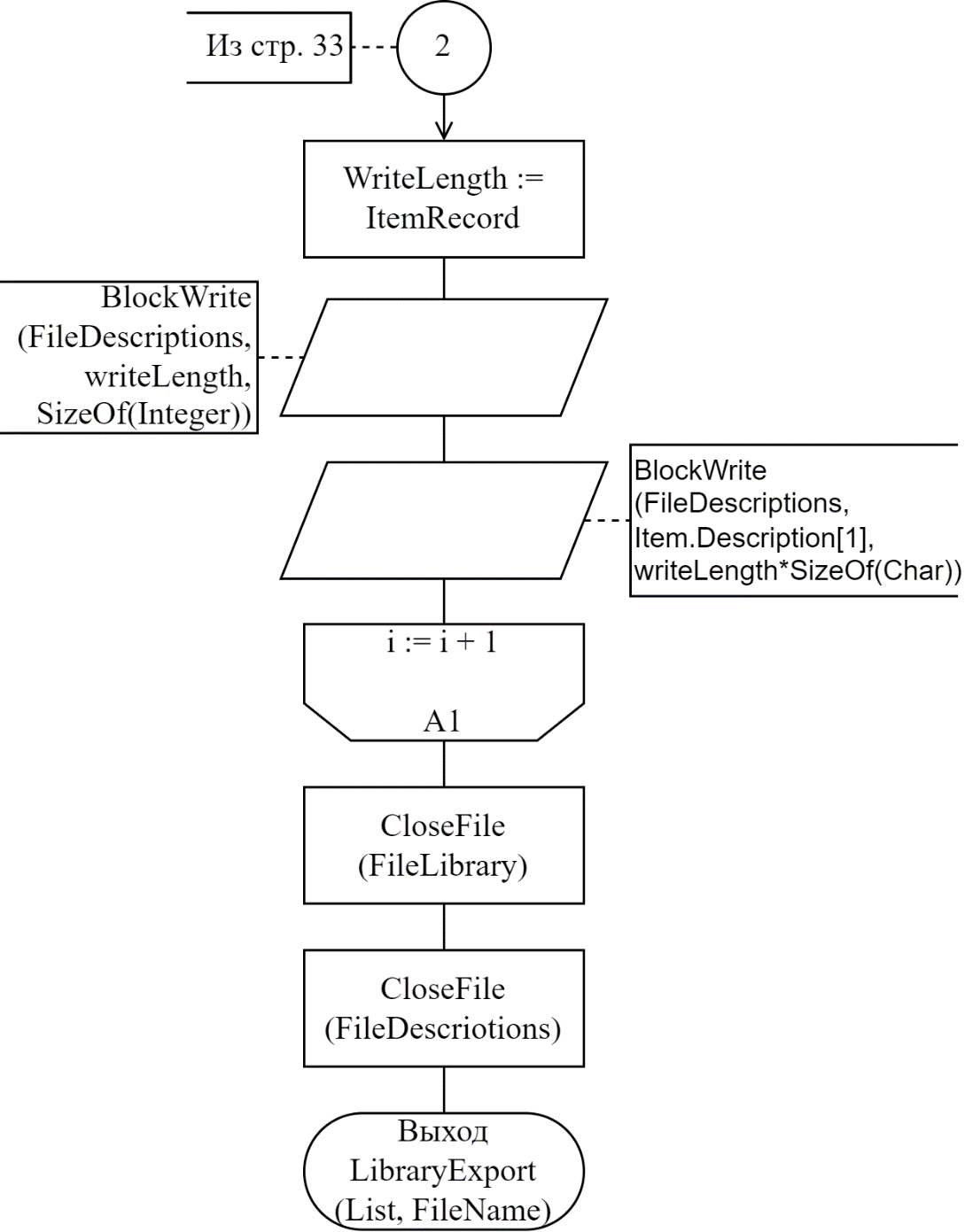
**

Рисунок 3.7 – Схема алгоритма LibraryExport (часть 3)

### Схема алгоритма LibraryImport

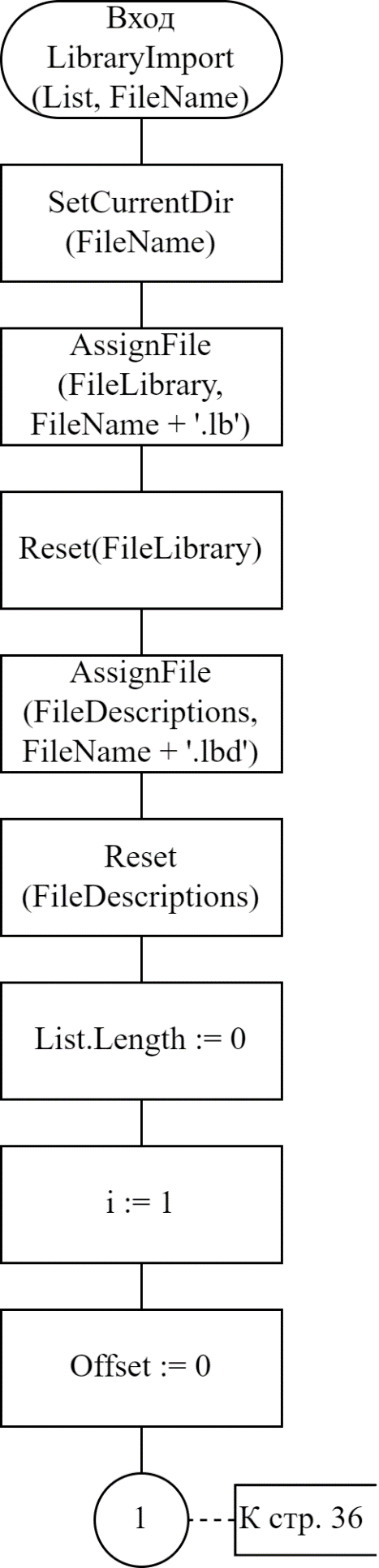


Рисунок 3.8 – Схема алгоритма LibraryImport (часть 1)

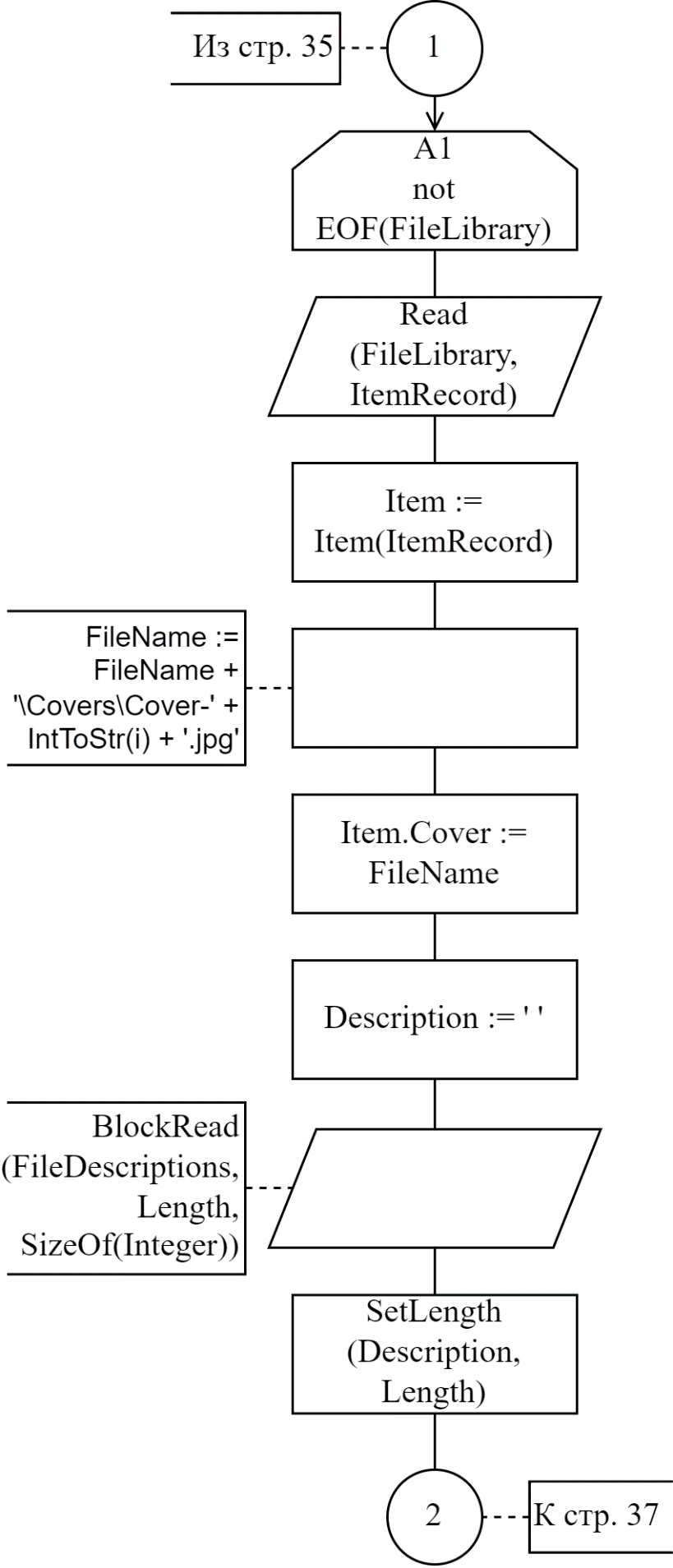


Рисунок 3.9 – Схема алгоритма LibraryImport (часть 2)

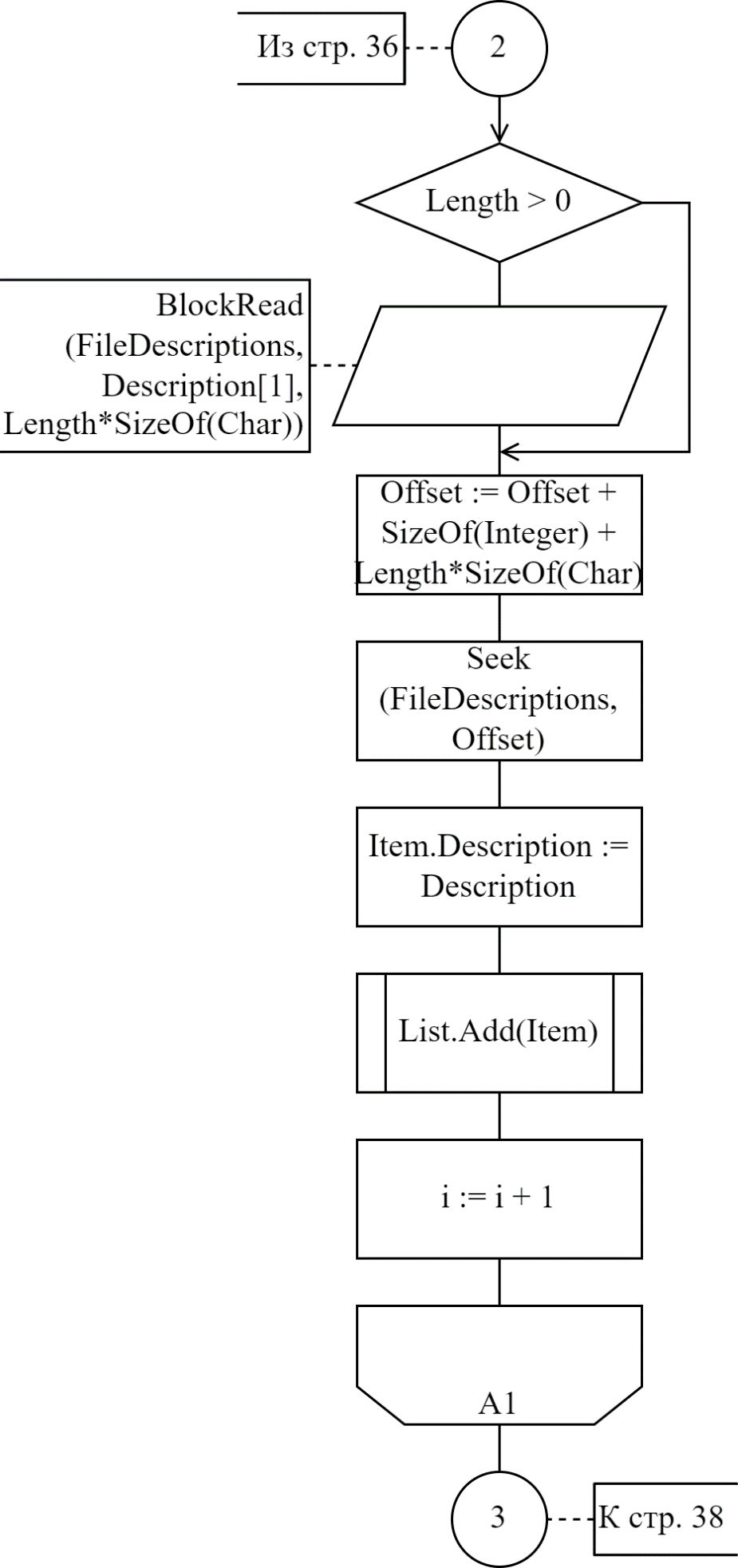


Рисунок 3.10 – Схема алгоритма LibraryImport (часть 3)

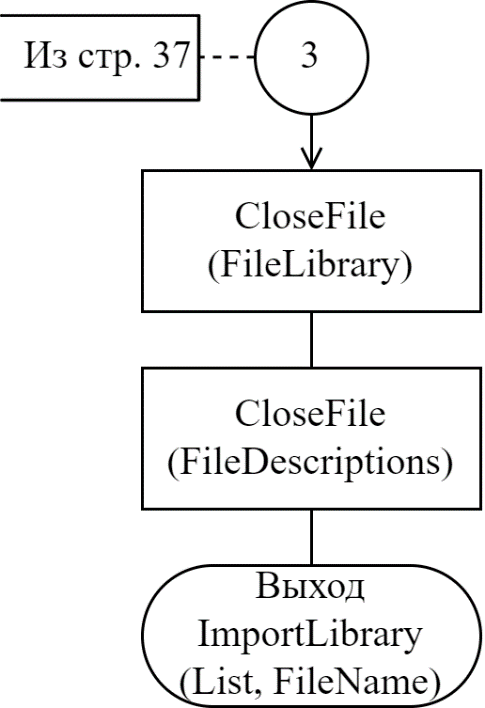


Рисунок 3.11 – Схема алгоритма LibraryImport (часть 4)

### Схема алгоритма SearchDown

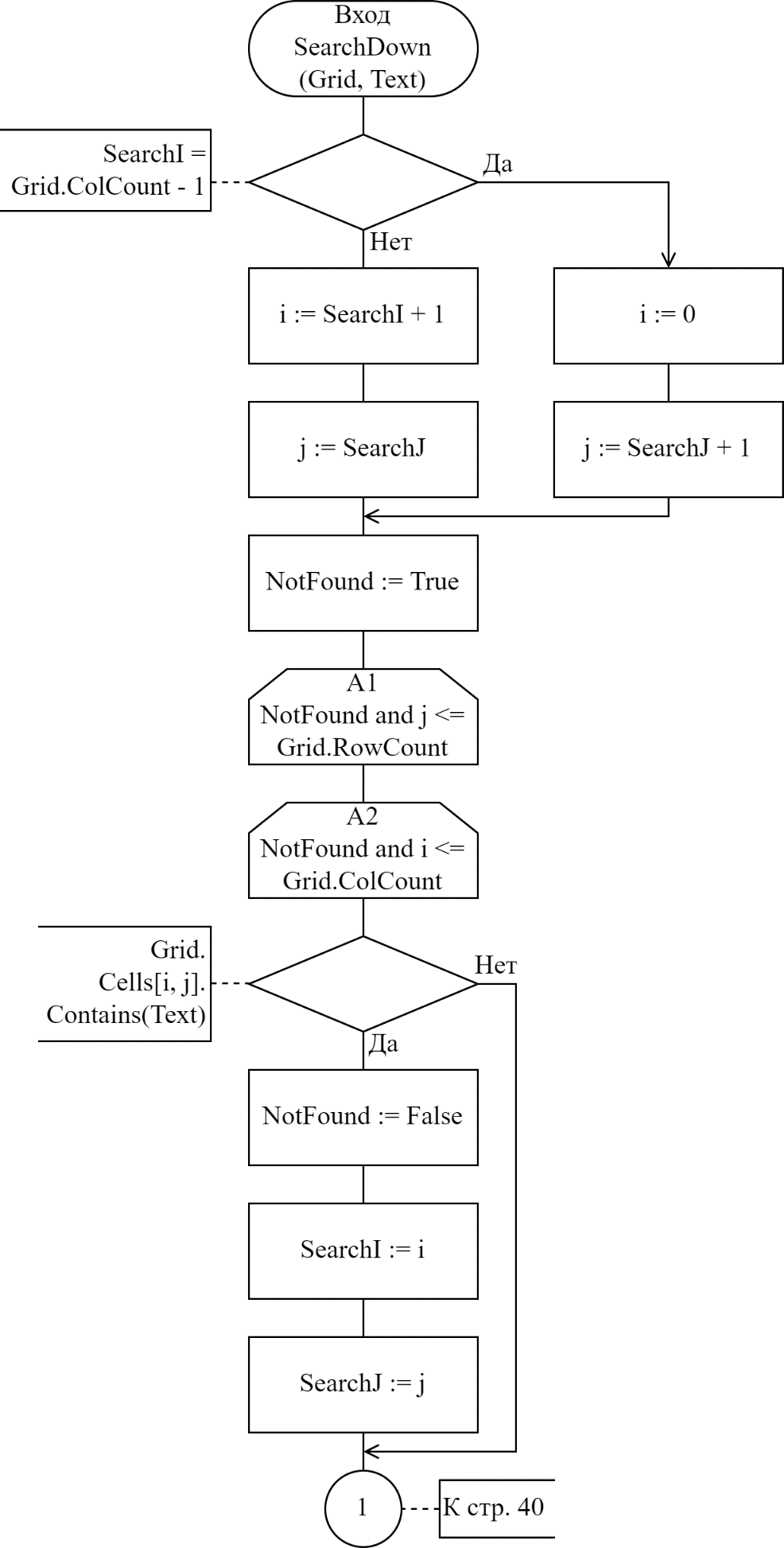


Рисунок 3.12 – Схема алгоритма SearchDown (часть 1)

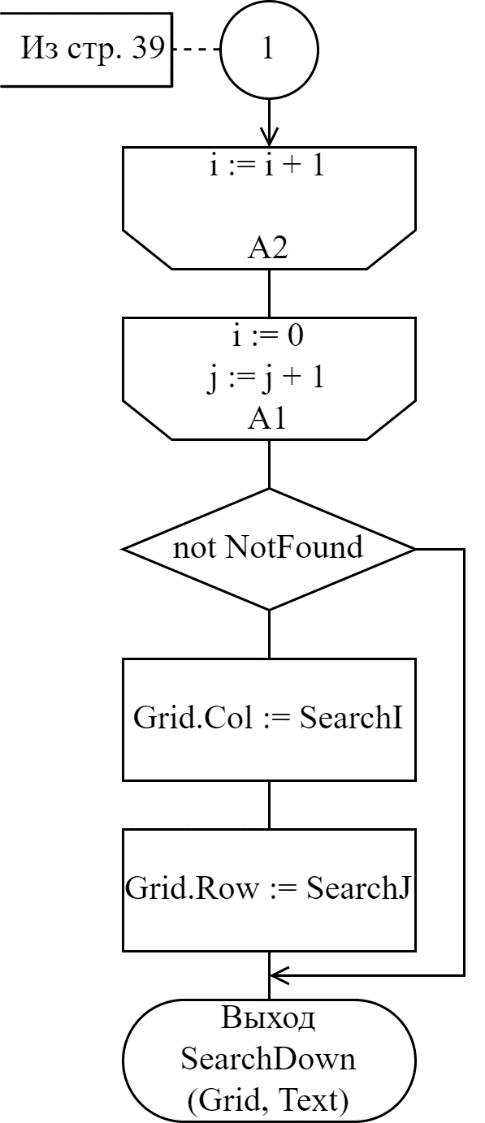


Рисунок 3.13 – Схема алгоритма SearchDown (часть 2)

### Схема алгоритма SearchUp

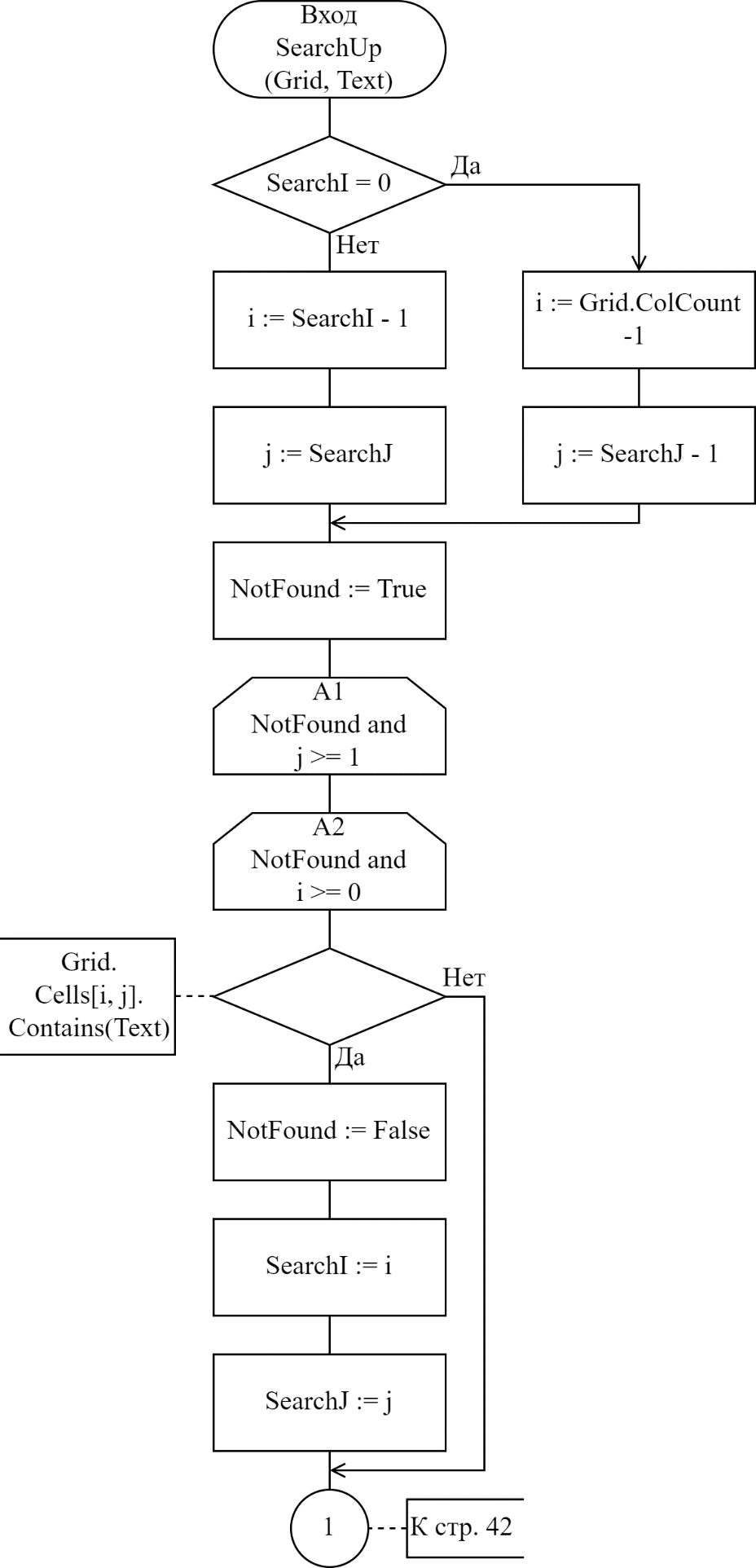


Рисунок 3.14 – Схема алгоритма SearchUp (часть 1)

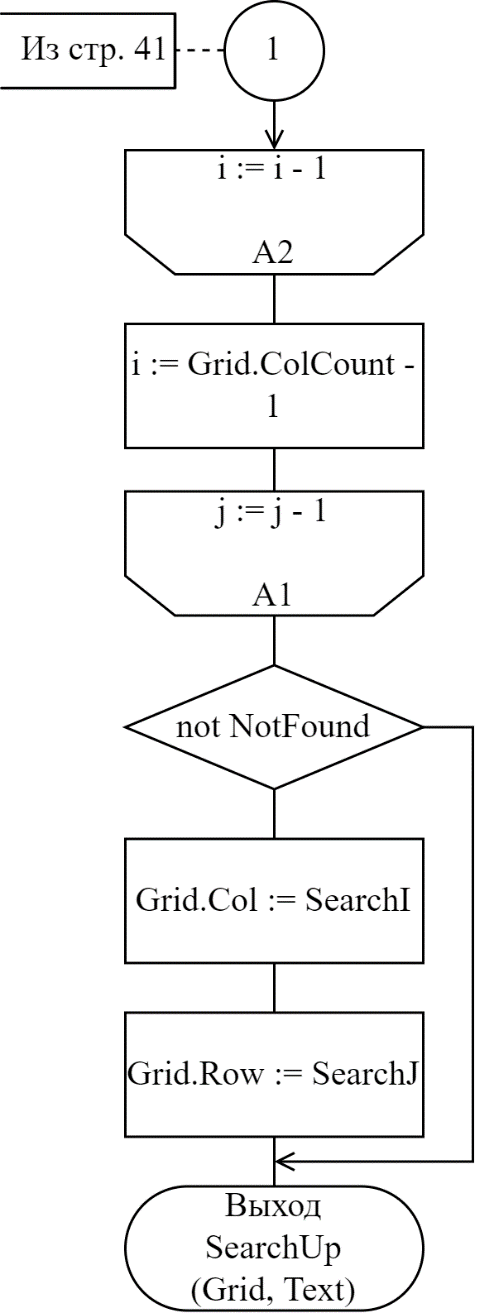


Рисунок 3.15 – Схема алгоритма SearchUp (часть 2)

### Схема алгоритма Swap

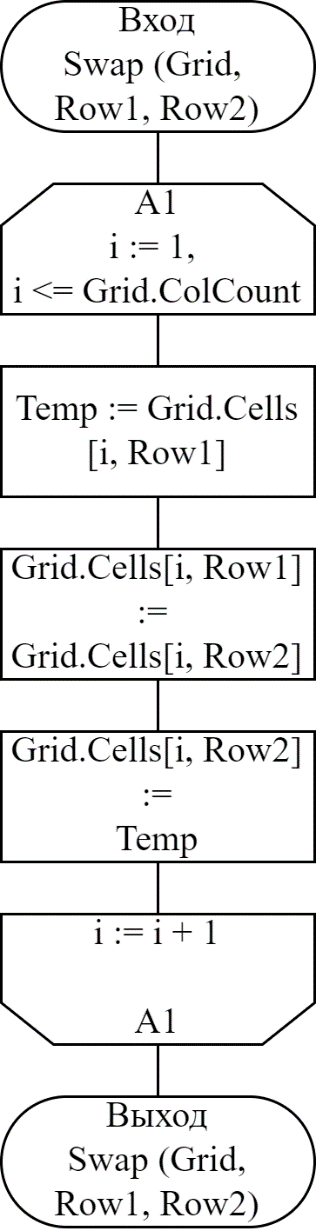


Рисунок 3.16 – Схема алгоритма Swap

### Схема алгоритма SortGridByCols

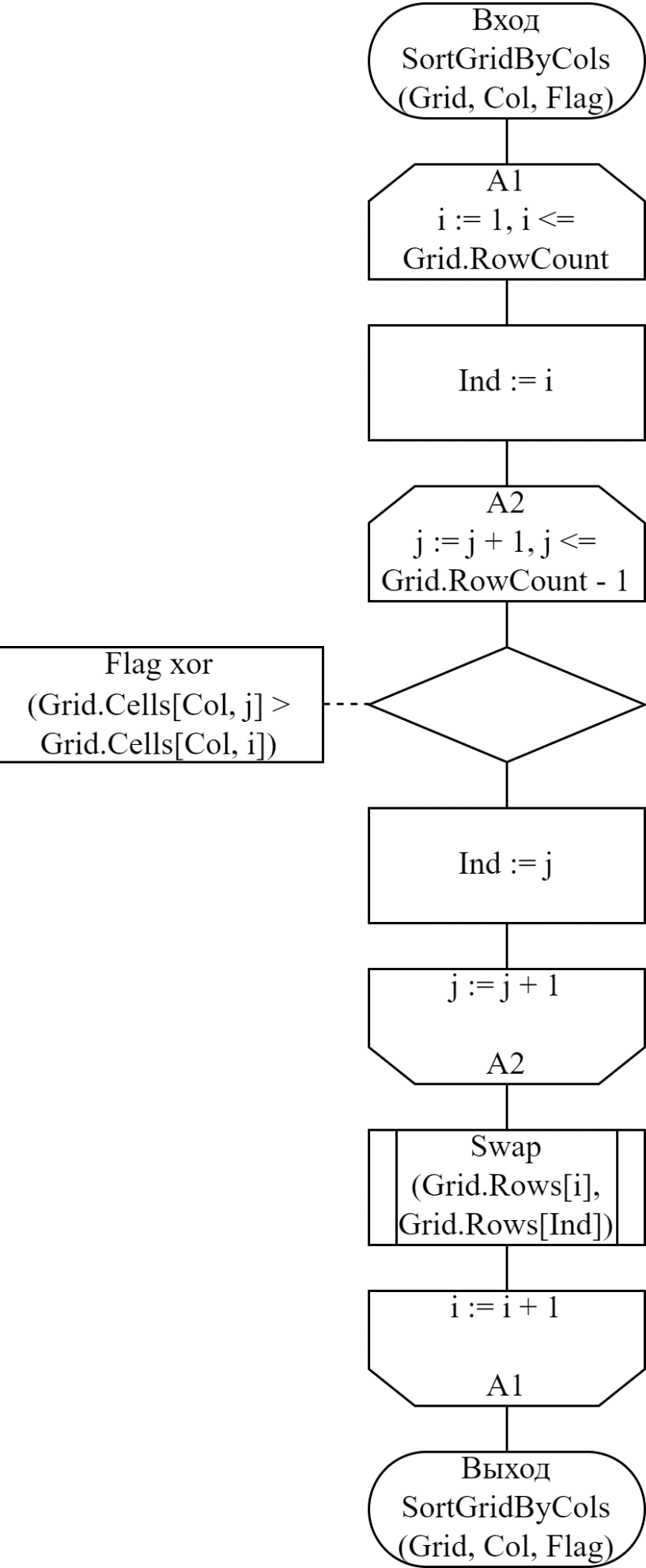


Рисунок 3.17 – Схема алгоритма SortGridByCols

### Схема алгоритма HelpTextSet

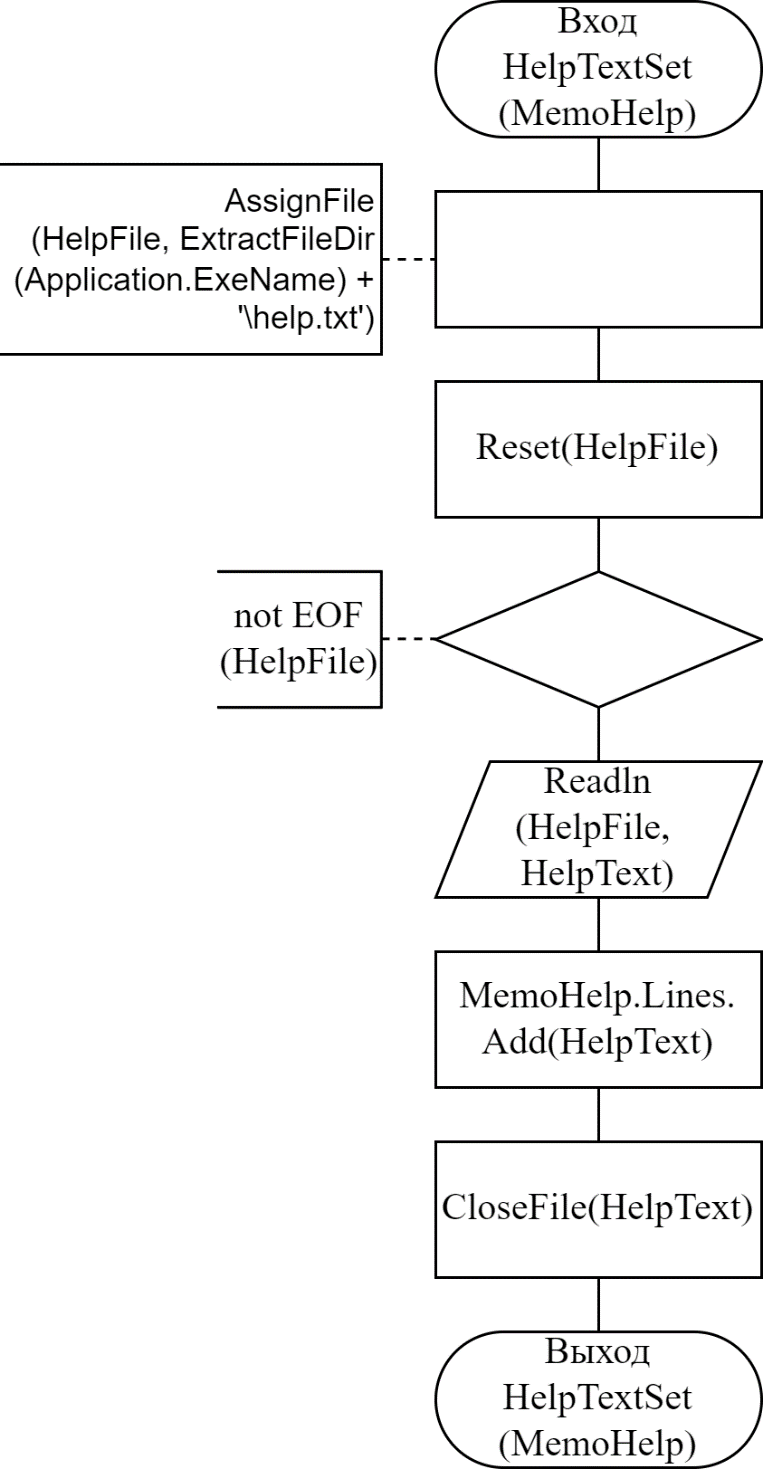


Рисунок 3.18 – Схема алгоритма HelpTextSet

# Тестирование и проверка работоспособности программного средства

## Запуск программного средства

### Тест 1

Таблица 4.1 – Тест 1

|  |  |
| --- | --- |
| Тестовая ситуация | Проверка корректного открытия формы при запуске программного средства |
| Исходный набор данных | Запуск программы |
| Ожидаемый результат | Корректное открытие основной формы |
| Полученный результат |  |

## Запуск окна добавления книг

### Тест 1

Таблица 4.2 – Тест 1

|  |  |
| --- | --- |
| Тестовая ситуация | Проверка корректного открытия формы при добавлении книг |
| Исходный набор данных | Запуск формы открытия книг |

Продолжение таблицы 4.2 – Тест 1

|  |  |
| --- | --- |
| Ожидаемый результат | Корректное открытие формы открытия книг |
| Полученный результат |  |

## Запуск окна добавления книг

### Тест 1

Таблица 4.3 – Тест 1

|  |  |
| --- | --- |
| Тестовая ситуация | Проверка корректного открытия формы при добавлении книг |
| Исходный набор данных | Запуск формы открытия книг |
| Ожидаемый результат | Корректное открытие формы открытия книг |
| Полученный результат |  |

## Запуск окна изменения книг

### Тест 1

Таблица 4.4 – Тест 1

|  |  |
| --- | --- |
| Тестовая ситуация | Проверка корректного открытия формы при изменении существующих книг |
| Исходный набор данных | Запуск формы изменения книг |
| Ожидаемый результат | Корректное открытие формы изменения книг |
| Полученный результат |  |

## Проверка ввода в числовое поле

### Тест 1

Таблица 4.5 – Тест 1

|  |  |
| --- | --- |
| Тестовая ситуация | Проверка корректного ввода в числовое поле количества страниц книги |
| Исходный набор данных | 13131313131313131313 |
| Ожидаемый результат | ‘13131313131313131313’ is not a valid integer value. |

Продолжение таблицы 4.5 – Тест 1

|  |  |
| --- | --- |
| Полученный результат |  |

## Проверка открытия диалоговых окон

### Тест 1

Таблица 4.6 – Тест 1

|  |  |
| --- | --- |
| Тестовая ситуация | Проверка корректного открытия диалогового окна сохранения библиотеки |
| Исходный набор данных | Запуск диалогового окна сохранения библиотеки |
| Ожидаемый результат | Корректное открытие диалогового окна сохранения библиотеки |
| Полученный результат |  |

### Тест 2

Таблица 4.7 – Тест 2

|  |  |
| --- | --- |
| Тестовая ситуация | Проверка корректного открытия диалогового окна открытия библиотеки |
| Исходный набор данных | Запуск диалогового окна открытия библиотеки |
| Ожидаемый результат | Корректное открытие диалогового окна открытия библиотеки |
| Полученный результат |  |

### Тест 3

Таблица 4.8 – Тест 3

|  |  |
| --- | --- |
| Тестовая ситуация | Проверка корректного открытия диалогового окна открытия обложки книги |
| Исходный набор данных | Запуск диалогового окна открытия обложки книги |
| Ожидаемый результат | Корректное открытие диалогового окна открытия обложки книги |
| Полученный результат |  |

## Запуск окна помощи

### Тест 1

Таблица 4.9 – Тест 1

|  |  |
| --- | --- |
| Тестовая ситуация | Проверка корректного открытия окна помощи |
| Исходный набор данных | Запуск окна помощи |
| Ожидаемый результат | Корректное открытие окна помощи |
| Полученный результат |  |

## Поиск книги

### Тест 1

Таблица 4.10 – Тест 1

|  |  |
| --- | --- |
| Тестовая ситуация | Проверка поиска первого найденного элемента |
| Исходный набор данных | Поиск ‘123’ |
| Ожидаемый результат | 123 |
| Полученный результат |  |

### Тест 2

Таблица 4.11 – Тест 2

|  |  |
| --- | --- |
| Тестовая ситуация | Проверка поиска элемента вниз |
| Исходный набор данных | Поиск ‘123’ вниз |
| Ожидаемый результат | 1234 |
| Полученный результат |  |

### Тест 3

Таблица 4.12 – Тест 3

|  |  |
| --- | --- |
| Тестовая ситуация | Проверка поиска элемента вверх |
| Исходный набор данных | Поиск ‘123’ вверх |

Продолжение таблицы 4.12 – Тест 3

|  |  |
| --- | --- |
| Ожидаемый результат | 123 |
| Полученный результат |  |

## Сортировка таблицы по столбцу

### Тест 1

Таблица 4.13 – Тест 1

|  |  |
| --- | --- |
| Тестовая ситуация | Проверка сортировки неотсортированной таблицы |
| Исходный набор данных | Сортировка по первому столбцу |
| Ожидаемый результат | (a, a, b, c, q, s) |
| Полученный результат |  |

### Тест 2

Таблица 4.14 – Тест 2

|  |  |
| --- | --- |
| Тестовая ситуация | Проверка сортировки отсортированной таблицы |
| Исходный набор данных | Сортировка по первому столбцу |
| Ожидаемый результат | (s, q, c, b, a, a) |
| Полученный результат |  |

### Тест 3

Таблица 4.15 – Тест 3

|  |  |
| --- | --- |
| Тестовая ситуация | Проверка сортировки отсортированной таблицы |
| Исходный набор данных | Сортировка по первому столбцу |
| Ожидаемый результат | (a, a, b, c, q, s) |

Продолжение таблицы 4.15 – Тест 3

|  |  |
| --- | --- |
| Полученный результат |  |

## Сохранение библиотеки

### Тест 1

Таблица 4.16 – Тест 1

|  |  |
| --- | --- |
| Тестовая ситуация | Проверка сохранения библиотеки |
| Исходный набор данных |  |
| Ожидаемый результат | Директория Library с файлами Library.lb, Library.lbd и директорией Covers с обложками книг Cover-1.jpg, Cover-2.jpg, Cover-3.jpg, Cover-4.jpg, Cover-5.jpg, Cover-6.jpg |

Продолжение таблицы 4.16 – Тест 1

|  |  |
| --- | --- |
| Полученный результат |  |

## Открытие библиотеки

### Тест 1

Таблица 4.17 – Тест 1

|  |  |
| --- | --- |
| Тестовая ситуация | Проверка сохранения библиотеки |
| Исходный набор данных |  |

Продолжение таблицы 4.17 – Тест 1

|  |  |
| --- | --- |
|  |  |
| Ожидаемый результат | (a, c, b, s, q, a) |
| Полученный результат |  |

# Руководство по установке и использованию программного средства

## Установка

На установочном диске находится установочный файл с программным средством. После открытия данного пакета на экране появляется окно, представленное на рисунке 5.1.

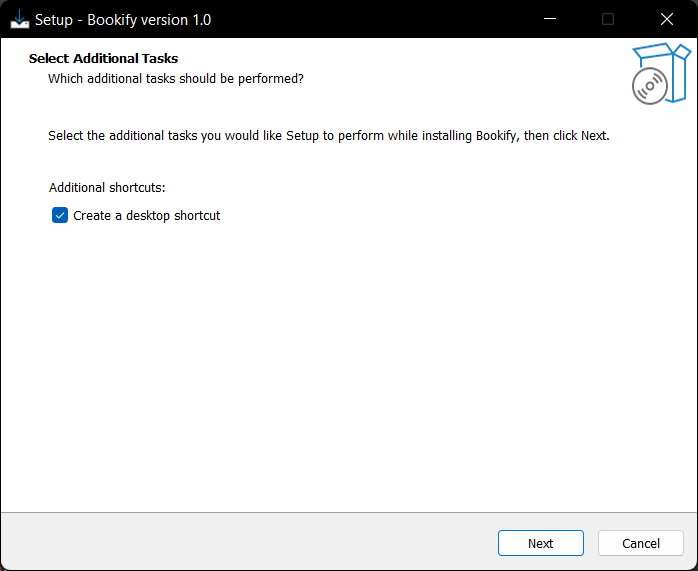


Рисунок 5.1 – Установка (этап 1)

После нажатия на «Далее» на экране открывается окно, представленное на рисунке 5.2. Здесь пользователь выбирает место папки установки.

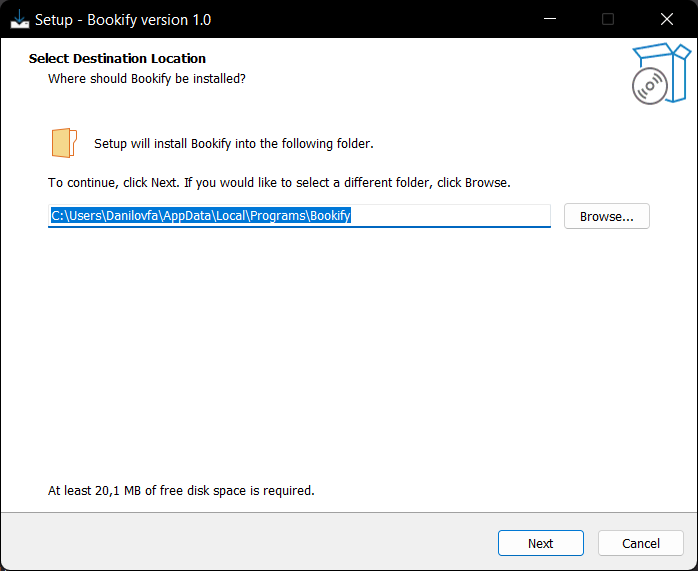


Рисунок 5.2 – Установка (этап 2)

После этого этапа подготовка к установке завершается и пользователю отображается предложение, показанное на рисунке 5.3, установить программное средство на компьютер.

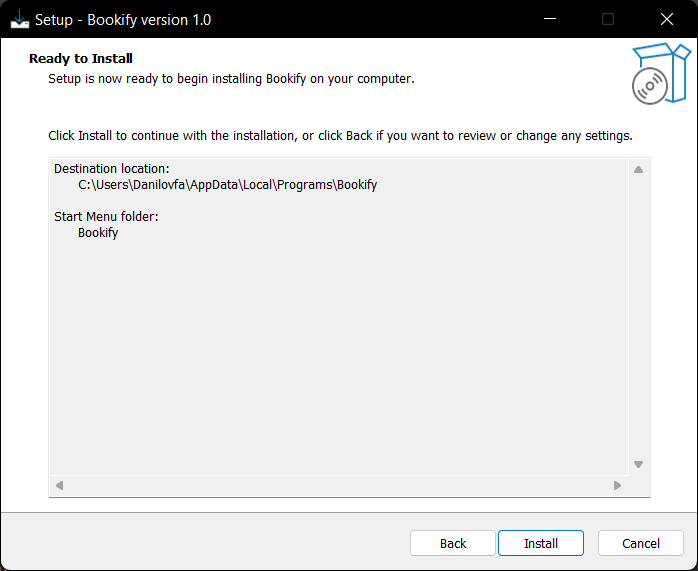


Рисунок 5.3 – Установка (этап 3)

Далее следует процесс установки, сопровождаемый визуализацией представленной на рисунке 5.4.

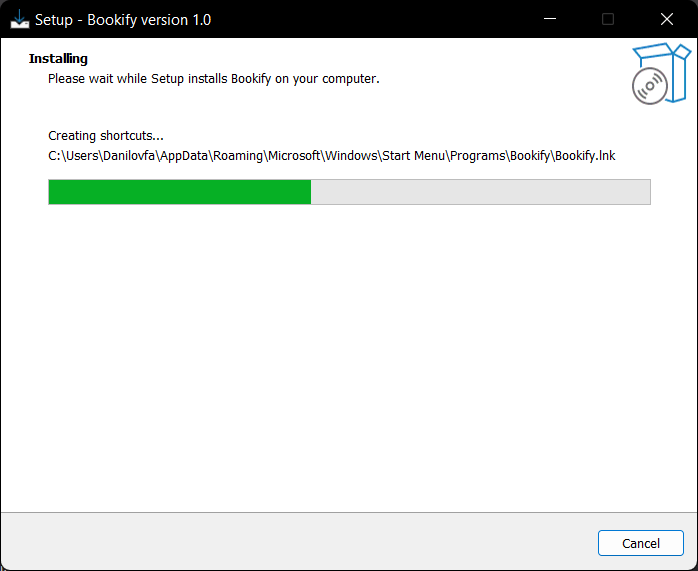


Рисунок 5.4 – Установка (этап 4)

Об успешном завершении установки пользователя информирует окно, представленное на рисунке 5.5.

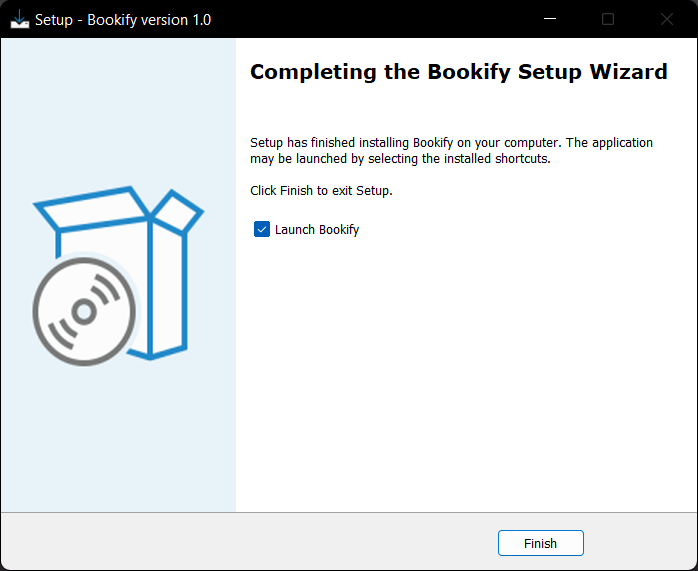


Рисунок 5.5 – Установка (этап 5)

## Использование

### Добавление книги

Добавить книгу можно посредством открытия окна, представленного ниже

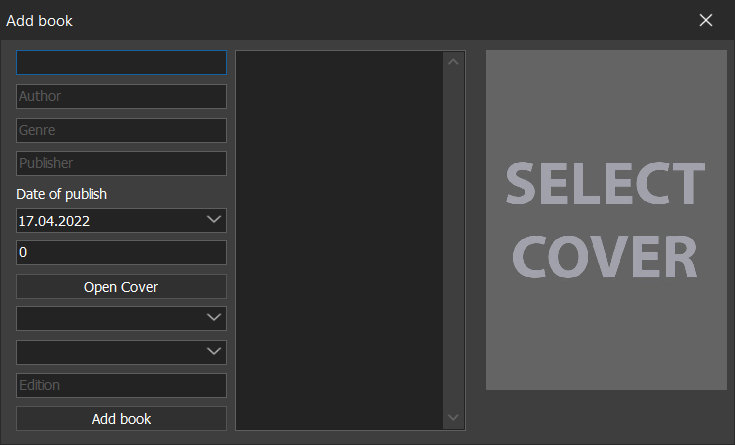


Рисунок 5.6 – Окно добавления книги

Вызвать данное окно можно тремя способами: нажатием на кнопку «Add book» в главном окне, нажатием на кнопку «Add book» в панели меню или нажатием комбинации клавиш Ctrl + N.

При нажатии на кнопку «Add book» в окне добавления книги окно закроется и книга добавится в таблицу на главном окне.

При нажатии на значок крестика в правом верхнем углу окно закроется.

### Изменение книги

Изменить книгу можно посредством открытия окна, представленного ниже

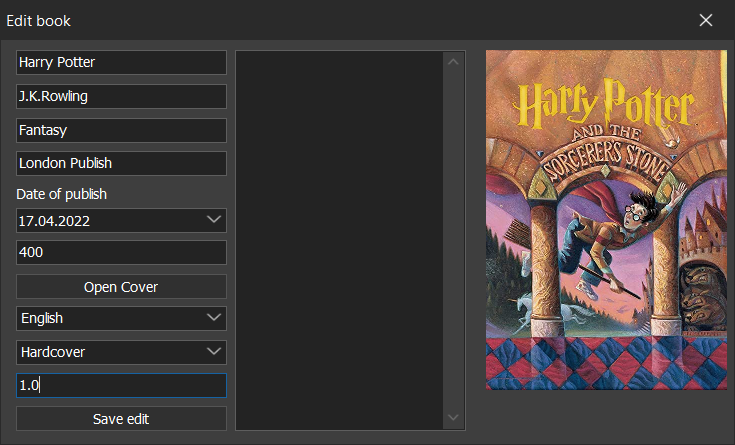


Рисунок 5.7 – Окно изменения книги

Вызвать данное окно можно тремя способами: нажатием на кнопку «Edit book» в главном окне, нажатием на кнопку «Edit book» в панели меню или нажатием комбинации клавиш Ctrl + E.

При открытии окна в поля заносятся данные выбранной книги в таблице в главном окне.

При нажатии на кнопку «Save edit» в окне изменения книги окно закроется и книга изменится в таблице на главном окне.

При нажатии на значок крестика в правом верхнем углу окно закроется.

### Сохранение библиотеки

Сохранить библиотеку можно посредством открытия диалогового окна сохранения файлов, представленного Embarcadero Delphi.

Вызвать данное окно можно тремя способами: нажатием на кнопку «Export Library» в главном окне, нажатием на кнопку «Export Library» в панели меню или нажатием комбинации клавиш Ctrl + S.

При нажатии на кнопку «Save» в окне сохранения файла окно закроется и библиотека сохранится на постоянную память компьютера.

При нажатии на значок крестика в правом верхнем углу или на кнопку «Cancel» окно закроется.

### Поиск книг

Найти книгу можно посредством открытия использования поля Search в главном окне программного средства.

При вводе текста в поле Search первый найденный элемент подсветится в таблице. Для дальнейшего поиска по таблице можно использовать стрелочки вверх и вниз.

### Сортировка таблицы

Отсортировать таблицу можно посредством двойного нажатия на заголовок столбца таблицы. При первом двойном нажатии таблица отсортируется по возрастанию, при дальнейших нажатиях сортировка таблицы будет меняться в обратную сторону.

# Заключение

По итогу работы над курсовым проектом было разработано программное средство «Домашняя библиотека», обладающее графическим интерфейсом для взаимодействия с пользователем.

Приложение прошло все этапы тестирования, в результате которых были устранены все неполадки.

Данное программное средство, кроме основных операций добавления, удаления и изменения книг предоставляет пользователю возможность поиска книг, сортировки библиотеки, сохранения библиотеки в файл, открытие библиотеки из файла, получение справки.

Бесценным результатом курсового проектирования является полученный опыт работы с динамической структурой данных и файлами. Был получен опыт работы с графическим интерфейсом.

При дальнейшей доработке программы, возможно добавление дополнительного функционала (работа с базами данных, сохранение и открытие файлов Excel и др.), а также улучшение производительности программы.

Список использованной литературы

[1] libib – Cloud Cataloging: [сайт]. URL: https://www.libib.com/

[2] CollectorZ Book Collector: [сайт]. URL: https://www.collectorz.com/book/book-collector

[3] Evergreen – Open Source Library Software [сайт]. URL: <https://evergreen-ils.org/>

[4] Глухова, Л.А. Основы алгоритмизации и программирования ч.2: учебное пособие / Л.А.Глухова – Минск: БГУИР, 2006 – 146 с.

[5] Вирт Н. Алгоритмы и структуры данных: учебное пособие / Н.Вирт – М.: Мир, 1989 – 126 с.

Приложение А

(обязательное)

Исходный код программы (модуль Main)

unit Main;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, System.IOUtils, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, System.Actions, Vcl.ActnList,

Vcl.ToolWin, Vcl.ActnMan, Vcl.ActnCtrls, Vcl.Menus, Vcl.ExtCtrls, Vcl.StdActns, Vcl.Grids, Vcl.ValEdit, Vcl.StdCtrls, Vcl.Buttons, Vcl.ComCtrls,

Vcl.JumpList, Data.DB, Vcl.DBGrids, Vcl.DBCtrls, Vcl.DBCGrids, Add, List, Database,

Vcl.Imaging.jpeg, Help, System.ImageList, Vcl.ImgList;

type

TMoveSG = class(TCustomGrid);

TForm\_Main = class(TForm)

MainMenu: TMainMenu;

ads1: TMenuItem;

Edit1: TMenuItem;

Help1: TMenuItem;

ActionList1: TActionList;

FileOpen1: TFileOpen;

FileExit1: TFileExit;

Book\_Add: TBitBtn;

Book\_Edit: TBitBtn;

Book\_Remove: TBitBtn;

Library\_Import: TBitBtn;

Library\_Export: TBitBtn;

Action\_Book\_Add: TAction;

Action\_Book\_Edit: TAction;

Action\_Database\_Import: TAction;

Action\_Database\_Export: TAction;

Addbook1: TMenuItem;

Action\_Book\_Remove: TAction;

Importbookdatabase1: TMenuItem;

Exportbookdatabase1: TMenuItem;

N1: TMenuItem;

Exit1: TMenuItem;

Editbook1: TMenuItem;

Removebook1: TMenuItem;

StringGrid: TStringGrid;

ImageCover: TImage;

Library\_OpenDialog: TOpenDialog;

Library\_SaveDialog: TSaveDialog;

MemoDescription: TMemo;

Edit\_Item: TEdit;

Action1: TAction;

MenuItemHelp: TMenuItem;

Edit\_Search: TEdit;

Button\_Up: TButton;

Button\_Down: TButton;

ImageList1: TImageList;

procedure Action\_Book\_EditExecute(Sender: TObject);

procedure Action\_Book\_AddExecute(Sender: TObject);

procedure FormCreate(Sender: TObject);

procedure UpdateGrid(List: TList);

procedure StringGridSelectCell(Sender: TObject; ACol, ARow: Integer;

var CanSelect: Boolean);

procedure StringGridDblClick(Sender: TObject);

// Sort string grid by column

procedure SortGridByCols(Grid: TStringGrid; Col: Integer; Flag: Boolean);

procedure Action\_Database\_ExportExecute(Sender: TObject);

procedure Action\_Database\_ImportExecute(Sender: TObject);

procedure StringGridDrawCell(Sender: TObject; ACol, ARow: Integer;

Rect: TRect; State: TGridDrawState);

procedure Action\_Book\_RemoveExecute(Sender: TObject);

procedure RemoveAttributes(StringGrid: TStringGrid);

procedure Action1Execute(Sender: TObject);

procedure Edit\_SearchChange(Sender: TObject);

procedure SearchUp(StringGrid: TStringGrid; Text: String);

procedure SearchDown(StringGrid: TStringGrid; Text: String);

procedure Button\_UpClick(Sender: TObject);

procedure Button\_DownClick(Sender: TObject);

private

{ Private declarations }

SearchI, SearchJ: Integer;

public

{ Public declarations }

Item: TItem;

List: TList;

end;

var

Form\_Main: TForm\_Main;

implementation

{$R \*.dfm}

// Procedure to run when form is created

procedure TForm\_Main.FormCreate(Sender: TObject);

begin

// Initialize List

List := TList.Create;

// Set Memo parent

MemoDescription.Parent := self;

MemoDescription.Lines.Clear;

// Set fixed row values

StringGrid.Cells[0,0] := 'Title';

StringGrid.Cells[1,0] := 'Author';

StringGrid.Cells[2,0] := 'Genre';

StringGrid.Cells[3,0] := 'Publisher';

StringGrid.Cells[4,0] := 'Date of publish';

StringGrid.Cells[5,0] := 'Pages';

StringGrid.Cells[6,0] := 'Language';

StringGrid.Cells[7,0] := 'Edition';

StringGrid.Cells[8,0] := 'Binding';

end;

procedure TForm\_Main.Action1Execute(Sender: TObject);

begin

with TFormHelp.Create(self) do

try

ShowModal;

finally

Free;

end;

end;

procedure TForm\_Main.Action\_Book\_AddExecute(Sender: TObject);

begin

with TFormAdd.AddCreate() do

// Open form

try

ShowModal;

finally

// Get item from created form

Self.Item := GetItem();

if Item.IsEmpty then

List.Add(Item);

Free;

end;

UpdateGrid(List);

RemoveAttributes(StringGrid);

end;

procedure TForm\_Main.Action\_Book\_EditExecute(Sender: TObject);

begin

if StringGrid.RowCount > 1 then

begin

with TFormAdd.EditCreate(List.GetById(StringGrid.Row-1)) do

// Open form

try

ShowModal;

finally

// Get item from created form

Self.Item := GetItem();

Free;

end;

// Update list entry

List.SetById(StringGrid.Row-1, Item);

// Update grid

UpdateGrid(List);

RemoveAttributes(StringGrid);

end;

end;

// If Row is not fixed remove it from list and update grid

procedure TForm\_Main.Action\_Book\_RemoveExecute(Sender: TObject);

begin

if StringGrid.Row > 0 then

begin

List.RemoveById(StringGrid.Row - 1);

UpdateGrid(List);

end;

RemoveAttributes(StringGrid);

end;

// Event when 'Export library' is clicked

procedure TForm\_Main.Action\_Database\_ExportExecute(Sender: TObject);

var

SaveDialog: TSaveDialog;

FileLibrary: File of TItemExport;

FileDescriptions: File;

NameLibrary: String;

Item: TItemExport;

ItemClass: TItem;

i, writeLength: Integer;

FileName, SavedCover: String;

begin

Library\_SaveDialog.Title := 'Save Library';

Library\_SaveDialog.InitialDir := GetCurrentDir;

if Library\_SaveDialog.Execute then

begin

// Create folder for library and move to it

TDirectory.CreateDirectory(Library\_SaveDialog.FileName);

SetCurrentDir(Library\_Savedialog.FileName);

NameLibrary := ExtractFileName(Library\_SaveDialog.FileName);

// Create library file

AssignFile(FileLibrary, NameLibrary +'.lb');

Rewrite(FileLibrary);

// Create descritions file

AssignFile(FileDescriptions, NameLibrary+'.lbd');

Rewrite(FileDescriptions, 1);

// FileDescriptions := TFile.Create(NameLibrary+'.lbd', fmCreate);

SavedCover := ExtractFilePath(Application.ExeName) + '\Cover.jpg';

// Create dir for covers

CreateDir('Covers');

// Iterate through list and add each record to file

for i := 1 to List.length do

begin

// Get Item from List

ItemClass := List.GetById(i-1);

Item := ItemClass.toRecord;

//

FileName := GetCurrentDir + '\Covers\Cover-' + IntToStr(i) + '.jpg';

// Save Cover to folder

if Item.Cover <> '' then

TFile.Copy(Item.Cover, FileName)

else

TFile.Copy(SavedCover, FileName);

// Append record to file

Item.Cover := FileName;

Write(FileLibrary, Item);

writeLength := ItemClass.Description.Length;

BlockWrite(FileDescriptions, writeLength, SizeOf(Integer));

BlockWrite(FileDescriptions, ItemClass.Description[1], writeLength\*SizeOf(Char));

// BlockWrite(FileDescriptions, writeLength, SizeOf(Integer));

// BlockWrite(FileDescriptions, ItemClass.Description, writeLength);

// FileDescriptions.Write(writeLength, SizeOf(Integer));

// FileDescriptions.Write(ItemClass.Description, writeLength);

end;

// Close file

CloseFile(FileLibrary);

CloseFile(FileDescriptions);

end;

end;

// Event when 'Import library' is clicked

procedure TForm\_Main.Action\_Database\_ImportExecute(Sender: TObject);

var

FileLibrary: File of TItemExport;

FileDescriptions: File;

Item: TItem;

ItemRecord: TItemExport;

i, Offset, Length: Integer;

FilePath, Description: String;

begin

// Configure Library\_OpenDialog

Library\_OpenDialog.Title := 'Open Library';

Library\_OpenDialog.InitialDir := GetCurrentDir;

Library\_OpenDialog.Filter := 'Library file|\*.lb';

// After opened

if Library\_OpenDialog.Execute then

begin

SetCurrentDir(ExtractFileDir(Library\_OpenDialog.FileName));

FilePath := ExtractFileName(Library\_OpenDialog.FileName);

// Open library file

AssignFile(FileLibrary, FilePath);

//FileMode := fmOpenRead;

Reset(FileLibrary);

// Open descriptions file

AssignFile(FileDescriptions, FilePath + 'd');

Reset(FileDescriptions, 1);

// Create new list

List := List.Create();

i := 1;

Offset := 0;

// Add records to list while not end of file

while not EOF(FileLibrary) do

begin

// Read to record

Read(FileLibrary, ItemRecord);

// Create Item and add record items to it

Item := TItem.Create;

Item.fromRecord(ItemRecord);

// Get path of image cover and add it to Item

FilePath := ExtractFileDir(Library\_OpenDialog.FileName)

+ '\Covers\Cover-' + IntToStr(i) + '.jpg';

Item.Cover := FilePath;

// Get Description and add it to Item

// Read Length of string

// Length := 0;

// Description := '';

BlockRead(FileDescriptions, Length, SizeOf(Integer));

//Set length of string

SetLength(Description, Length);

// Read string

if Length > 0 then

BlockRead(FileDescriptions, Description[1], Length\*SizeOf(Char));

Offset := Offset + SizeOf(Integer) + Length\*SizeOf(Char);

Seek(FileDescriptions, Offset);

Item.Description := Description;

// Add Item to List

List.Add(Item);

i := i + 1;

end;

// Close files

CloseFile(FileLibrary);

CloseFile(FileDescriptions);

// Update grid after importing list

UpdateGrid(List);

end;

end;

procedure TForm\_Main.Button\_DownClick(Sender: TObject);

begin

SearchDown(StringGrid, Edit\_Search.Text);

end;

procedure TForm\_Main.Button\_UpClick(Sender: TObject);

begin

SearchUp(StringGrid, Edit\_Search.Text);

end;

procedure TForm\_Main.Edit\_SearchChange(Sender: TObject);

begin

SearchI := -1;

SearchJ := 1;

SearchDown(StringGrid, Edit\_Search.Text);

end;

procedure TForm\_Main.SearchUp(StringGrid: TStringGrid; Text: String);

var

i, j: Integer;

NotFound: Boolean;

begin

if SearchI = 0 then

begin

i := StringGrid.ColCount - 1;

j := SearchJ - 1;

end else

begin

i := SearchI - 1;

j := SearchJ;

end;

NotFound := True;

while (NotFound) and (j >= 1) do

begin

while (NotFound) and (i >= 0) do

begin

if StringGrid.Cells[i, j].Contains(Text) then

begin

NotFound := False;

SearchI := i;

SearchJ := j;

end;

i := i - 1;

end;

i := StringGrid.ColCount - 1;

j := j - 1;

end;

if not NotFound then

begin

StringGrid.Col := SearchI;

StringGrid.Row := SearchJ;

end;

end;

procedure TForm\_Main.SearchDown(StringGrid: TStringGrid; Text: String);

var

i, j: Integer;

NotFound: Boolean;

begin

if SearchI = StringGrid.ColCount - 1 then

begin

i := 0;

j := SearchJ + 1;

end else

begin

i := SearchI + 1;

j := SearchJ;

end;

NotFound := True;

while (NotFound) and (j <= StringGrid.RowCount) do

begin

while (NotFound) and (i <= StringGrid.ColCount) do

begin

if StringGrid.Cells[i, j].Contains(Text) then

begin

NotFound := False;

SearchI := i;

SearchJ := j;

end;

i := i + 1;

end;

i := 0;

j := j + 1;

end;

if not NotFound then

begin

StringGrid.Col := SearchI;

StringGrid.Row := SearchJ;

end;

end;

// Change MemoDescription and ImageCover when cell is selected

procedure TForm\_Main.StringGridSelectCell(Sender: TObject; ACol, ARow: Integer;

var CanSelect: Boolean);

var

Item: TItem;

begin

Edit\_Item.Text := StringGrid.Cells[ACol, ARow];

if (ARow > 0) then

begin

// Get record of book

Item := List.GetById(ARow-1);

// Set description

MemoDescription.Lines.Clear;

MemoDescription.Lines.Text := Item.Description;

// Set ImageCover

if Item.Cover <> '' then

ImageCover.Picture.LoadFromFile(Item.Cover)

else

self.ImageCover.Picture.LoadFromFile(ExtractFilePath(Application.ExeName) + '\Cover.jpg');

end

else

begin

MemoDescription.Text := 'Select book';

self.ImageCover.Picture.LoadFromFile(ExtractFilePath(Application.ExeName) + '\Cover.jpg');

end;

end;

// Update grid from list

procedure TForm\_Main.UpdateGrid(List: TList);

var

i, j: Integer;

Item: TItem;

begin

StringGrid.RowCount := 1;

for i := 1 to List.length do

begin

StringGrid.RowCount := StringGrid.RowCount + 1;

Item := List.GetById(i-1);

StringGrid.Cells[0,i] := Item.Title;

StringGrid.Cells[1,i] := Item.Author;

StringGrid.Cells[2,i] := Item.Genre;

StringGrid.Cells[3,i] := Item.Publisher;

StringGrid.Cells[4,i] := DateToStr(Item.Date);

StringGrid.Cells[5,i] := IntToStr(Item.Pages);

StringGrid.Cells[6,i] := Item.Language;

StringGrid.Cells[7,i] := Item.Edition;

StringGrid.Cells[8,i] := Item.Binding;

end;

end;

// Sort when stringgrid header is double-clicked

procedure TForm\_Main.StringGridDblClick(Sender: TObject);

var

Cell: String;

begin

if StringGrid.Row = 0 then

begin

Cell := StringGrid.Cells[StringGrid.Col, StringGrid.Row];

if Cell.Contains('(+)') then

begin

RemoveAttributes(StringGrid);

Cell := Cell.Remove(Cell.Length - 4);

Cell := Cell + ' (-)';

SortGridByCols(StringGrid, StringGrid.Col, False);

end

else if Cell.Contains('(-)') then

begin

RemoveAttributes(StringGrid);

Cell := Cell.Remove(Cell.Length - 4);

Cell := Cell + ' (+)';

SortGridByCols(StringGrid, StringGrid.Col, True);

end

else

begin

RemoveAttributes(StringGrid);

Cell := Cell + ' (+)';

SortGridByCols(StringGrid, StringGrid.Col, True);

end;

StringGrid.Cells[StringGrid.Col, StringGrid.Row] := Cell;

Edit\_Item.Text := Cell;

end;

end;

procedure TForm\_Main.StringGridDrawCell(Sender: TObject; ACol, ARow: Integer;

Rect: TRect; State: TGridDrawState);

begin

// bebra

end;

procedure TForm\_Main.SortGridByCols(Grid: TStringGrid; Col: Integer; Flag: Boolean);

var

i, j, MaxInd: Integer;

procedure Swap(Row1, Row2: Integer);

var

Temp: String;

i: Integer;

begin

for i := 0 to Grid.ColCount do

begin

Temp := StringGrid.Cells[i, Row1];

Grid.Cells[i, Row1] := Grid.Cells[i, Row2];

Grid.Cells[i, Row2] := Temp;

end;

end;

begin

for i := 1 to StringGrid.RowCount do

begin

MaxInd := i;

for j := i + 1 to StringGrid.RowCount-1 do

if Flag xor (Grid.Cells[Col, j] > Grid.Cells[Col, i]) then

maxInd := j;

Swap(i, MaxInd);

end;

Grid.Refresh;

end;

procedure TForm\_Main.RemoveAttributes(StringGrid: TStringGrid);

var

i: Integer;

Cell: String;

begin

for i := 0 to StringGrid.ColCount do

begin

Cell := StringGrid.Cells[i, StringGrid.Row];

if Cell.EndsWith(')') then

Cell := Cell.Remove(Cell.Length - 4);

StringGrid.Cells[i, StringGrid.Row] := Cell;

end;

end;

end.

Приложение Б

(обязательное)

Исходный код программы (модуль Add)

unit Add;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Vcl.ComCtrls, Vcl.Menus,

System.Actions, Vcl.ActnList, Vcl.ExtDlgs, Vcl.StdActns, Vcl.ExtCtrls, Database,

Vcl.Imaging.jpeg;

type

TFormAdd = class(TForm)

EditTitle: TEdit;

EditAuthor: TEdit;

EditGenre: TEdit;

EditPublisher: TEdit;

EditPages: TEdit;

EditDate: TDateTimePicker;

TextDate: TStaticText;

ButtonOpenCover: TButton;

ActionList1: TActionList;

FileOpen1: TFileOpen;

ImageCover: TImage;

OpenPictureDialog1: TOpenPictureDialog;

ListLang: TComboBox;

EditEdition: TEdit;

ButtonAdd: TButton;

ListBinding: TComboBox;

MemoDescription: TMemo;

procedure ButtonOpenCoverClick(Sender: TObject);

procedure FormCreate(Sender: TObject);

procedure ButtonAddClick(Sender: TObject);

function GetItem(): TItem;

constructor EditCreate(Item: TItem);

constructor AddCreate();

private

{ Private declarations }

public

{ Public declarations }

Item: TItem;

end;

var

FormAdd: TFormAdd;

implementation

{$R \*.dfm}

procedure TFormAdd.ButtonOpenCoverClick(Sender: TObject);

begin

if OpenPictureDialog1.Execute then

ImageCover.Picture.LoadFromFile(OpenPictureDialog1.FileName);

end;

procedure TFormAdd.ButtonAddClick(Sender: TObject);

begin

Item.Title := EditTitle.Text;

Item.Author := EditAuthor.Text;

Item.Genre := EditGenre.Text;

Item.Description := MemoDescription.Lines.Text;

Item.Publisher := EditPublisher.Text;

Item.Date := EditDate.Date;

Item.Pages := StrToInt(EditPages.Text);

Item.Cover := OpenPictureDialog1.FileName;

Item.Language := ListLang.Text;

Item.Edition := EditEdition.Text;

Item.Binding := ListBinding.Text;

Close;

end;

procedure TFormAdd.FormCreate(Sender: TObject);

begin

Item := TItem.Create;

// ImageCover.Picture.LoadFromFile('\Covers\Cover.jpg');

// Item.Cover := '\Covers\Cover.jpg';

end;

function TFormAdd.GetItem(): TItem;

begin

Result := Item;

end;

constructor TFormAdd.EditCreate(Item: TItem);

begin

self := TFormAdd.Create(nil);

self.Caption := 'Edit book';

self.ButtonAdd.Caption := 'Save edit';

EditPages.NumbersOnly := true;

self.EditTitle.Text := Item.Title;

self.EditAuthor.Text := Item.Author;

self.EditGenre.Text := Item.Genre;

self.EditPublisher.Text := Item.Publisher;

self.EditPages.Text := IntToStr(Item.Pages);

self.EditDate.Date := Item.Date;

if Item.Cover <> '' then

self.ImageCover.Picture.LoadFromFile(Item.Cover)

else

self.ImageCover.Picture.LoadFromFile(ExtractFilePath(Application.ExeName) + '\Cover.jpg');

self.MemoDescription.Lines.Text := Item.Description;

self.EditEdition.Text := Item.Edition;

end;

constructor TFormAdd.AddCreate();

begin

self := TFormAdd.Create(nil);

self.Caption := 'Add book';

MemoDescription.Lines.Clear;

EditPages.NumbersOnly := true;

self.ImageCover.Picture.LoadFromFile(ExtractFilePath(Application.ExeName) + '\Cover.jpg');

end;

end.

Приложение В

(обязательное)

Исходный код программы (модуль Help)

unit Help;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls;

type

TFormHelp = class(TForm)

MemoHelp: TMemo;

procedure FormCreate(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

FormHelp: TFormHelp;

implementation

{$R \*.dfm}

procedure TFormHelp.FormCreate(Sender: TObject);

var

HelpFile: TextFile;

HelpText: String;

begin

MemoHelp.Lines.Clear;

AssignFile(HelpFile, ExtractFileDir(Application.ExeName) + '\help.txt');

Reset(HelpFile);

while not EOF(HelpFile) do

begin

readln(HelpFile, HelpText);

MemoHelp.Lines.Add(HelpText);

end;

CloseFile(HelpFile);

end;

end.

Приложение Г

(обязательное)

Исходный код программы (модуль List)

unit List;

interface

uses vcl.Graphics, System.SysUtils, Database;

type

PNode=^TNode;

TNode = Record

data:TItem;

next:PNode;

End;

TList = class

private

// Pointer to the first node of TList

head:PNode;

public

// Property that stores total number

// of nodes in TList

length:Integer;

// Initialize new instance of TList

constructor Create(length: Integer); overload;

// Initialize new instance of TList

constructor Create; overload;

// Set data to element by id

procedure SetById(id:Integer; val:TItem);

// Remove node by Id

procedure RemoveById(id: Integer);

// Return data by id

function GetById(id: Integer): TItem;

// Add element to the back

procedure Add(data: TItem);

end;

implementation

constructor TList.Create;

begin

self.length := 0;

end;

constructor TList.Create(length: Integer);

var first, second: PNode;

i: Integer;

begin

// Set length property

self.length := length;

// Initialize head

new(first);

first.data := TItem.Create;

self.head := nil;

self.head := first;

// Initialize second element and set pointers

new(second);

head.next := second;

second.data := TItem.Create;

// Get ready for loop

first := second;

// Create new nodes from 3 to length

for i := 3 to length do

begin

// Initialize new node and set fields

new(second);

second.data := TItem.Create;

first.next := second;

// Get ready for next iteration (t := t + 1)

first := second;

end;

// Set pointers from/to head

first.next := nil;

end;

procedure TList.SetById(id:Integer; val:TItem);

var

Node:PNode;

i:Integer;

begin

// Move pointer to element

Node := self.head;

for i := 0 to id - 1 do

begin

Node := Node.next;

end;

// Set data

Node.data := val;

end;

procedure TList.RemoveById(id: Integer);

var

el1, el2: PNode;

i: Integer;

begin

if self.length = 1 then

begin

self.head := nil;

self.length := 0;

exit;

end;

if id = 0 then

begin

self.head := self.head.next;

self.length := self.length - 1;

exit;

end;

// Move cursor to desired node

el1 := self.head;

for i := 0 to id - 2 do

el1 := el1.next;

el2 := el1.next;

// If head is removed set self.head

// pointer to new location

// if el2 = self.head then

// self.head := el2.next;

// Remove desired node

el1.next := el2.next;

// Set values of deleted node to 0

el2 := nil;

// Change length property

self.length := self.length - 1;

end;

function TList.GetById(id: Integer): TItem;

var

el: PNode;

i: Integer;

begin

// Move to desired node

el := self.head;

for i := 0 to id - 1 do

el := el.next;

// Return data of desired node

Result := el.data;

end;

procedure TList.Add(data: TItem);

var

nodeNew, node: PNode;

begin

// Create new node

New(nodeNew);

nodeNew.data := data;

// If there are no nodes make new node head

if self.length = 0 then

begin

self.head := nodeNew;

self.head.next := nil;

end

else

begin

node := self.head;

while(node.next <> nil) do

node := node.next;

node.next := nodeNew;

nodeNew.next := nil;

end;

self.length := self.length + 1;

end;

end.

Приложение Д

(обязательное)

Исходный код программы (модуль Database)

unit Database;

interface

uses vcl.Graphics, system.SysUtils, system.Classes;

type

TItemExport = record

Title, Author, Genre, Publisher, Language, Edition, Binding: String[255];

Date: TDate;

Pages: Integer;

Cover: String[255];

end;

TItem = class

Title, Author, Genre, Publisher, Language, Edition, Binding: String;

Description: String;

Date: TDate;

Pages: Integer;

Cover: String;

constructor Create;

public

function IsEmpty: Boolean;

function toRecord(): TItemExport;

procedure fromRecord(ItemRecord: TItemExport);

End;

implementation

constructor TItem.Create;

begin

self.Title := '';

self.Author := '';

self.Genre := '';

self.Description := '';

self.Publisher := '';

self.Date := 0;

self.Pages := 0;

self.Cover := '';

self.Language := '';

self.Edition := '';

self.Binding := '';

end;

function TItem.toRecord(): TItemExport;

begin

Result.Title := self.Title;

Result.Author := self.Author;

Result.Genre := self.Genre;

Result.Publisher := self.Publisher;

Result.Language := self.Language;

Result.Edition := self.Edition;

Result.Binding := self.Binding;

Result.Date := self.Date;

Result.Pages := self.Pages;

Result.Cover := self.Cover;

end;

procedure TItem.fromRecord(ItemRecord: TItemExport);

begin

self.Title := ItemRecord.Title;

self.Author := ItemRecord.Author;

self.Genre := ItemRecord.Genre;

self.Publisher := ItemRecord.Publisher;

self.Language := ItemRecord.Language;

self.Edition := ItemRecord.Edition;

self.Binding := ItemRecord.Binding;

self.Date := ItemRecord.Date;

self.Pages := ItemRecord.Pages;

self.Cover := ItemRecord.Cover;

end;

function TItem.IsEmpty: Boolean;

begin

Result := false;

if self.Title <> '' then

Result := true;

if self.Author <> '' then

Result := true;

if self.Genre <> '' then

Result := true;

if self.Publisher <> '' then

Result := true;

if self.Language <> '' then

Result := true;

if self.Edition <> '' then

Result := true;

if self.Binding <> '' then

Result := true;

if DateToStr(self.Date) <> '' then

Result := true;

if self.Pages <> 0 then

Result := true;

if self.Description <> '' then

Result := true;

if self.Cover <> '' then

end;

end.