МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ «ВИТЕБСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ П.М. МАШЕРОВА»

Факультет математики и информационных технологий

Кафедра информационных технологий и управления бизнесом

КУРСОВАЯ РАБОТА

по дисциплине «Операционные системы и системное программирование»

**разработка консольного приложения для ОС «Windows» «Файловый менеджер»**

Верхолазов Никита Юрьевич,

3 курс, группа 34з

Руководитель:

Шпаков Сергей Андреевич,

**Старший преподаватель**

Витебск 2022

РЕФЕРАТ

Отчет по курсовой работе: с. 28, рис 7, ист. 5, прил. 2.

ПРОГРАММНЫЙ ПРОДУКТ, СРУКТУРА, ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ, ПРОЕКТ, ИНТЕРФЕЙС, СРЕДСТВА РАЗРАБОТКИ, ПРОЕКТИРОВАНИЕ, ТЕСТИРОВАНИЕ.

Целью работы является разработка консольного приложения для ОС «Windows» «Файловый менеджер».

Объектом и предметом исследования являются: язык программирования «C#», работа с файловой системой ОС «Windows» на языке программирования «C#».

В первом разделе содержится описание предметной области и описание используемых технологий.

Во втором разделе описывается практическая реализация проекта.

Автор проекта подтверждает, что приведённый в нём расчётно-аналитический материал правильно и объективно отражает состояние исследуемого процесса, а все заимствованные из литературных и других источников теоретические и методологические положения и концепции сопровождаются ссылками на их автора

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 4](#_Toc95070666)

[1 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ 5](#_Toc95070667)

[1.1 Язык программирования C# 5](#_Toc95070668)

[1.2 Устройство файловой системы ОС «Windows» 6](#_Toc95070669)

[1.3 Работа с файловой системой в языке программирования C# 9](#_Toc95070670)

[1.3.1 Работа с дисками 9](#_Toc95070671)

[1.3.2 Работа с каталогами 10](#_Toc95070672)

[1.3.3 Работа с файлами 11](#_Toc95070673)

[2 ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ ПРОЕКТА 13](#_Toc95070674)

[2.1 Формирование структуры проекта 13](#_Toc95070675)

[2.2 Алгоритмическое обеспечение проекта 13](#_Toc95070676)

[2.3 Реализация программного продукта 13](#_Toc95070677)

[ЗАКЛЮЧЕНИЕ 19](#_Toc95070678)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 20](#_Toc95070679)

[ПРИЛОЖЕНИЕ А 21](#_Toc95070680)

[ПРИЛОЖЕНИЕ Б 28](#_Toc95070681)

# ВВЕДЕНИЕ

Уже несколько десятков лет компьютер занял большую часть нашей повседневной жизни, и многие не представляют своей учёбы, работы без компьютера. В настоящее время трудно назвать те области человеческой деятельности, успехи в которых не были бы связаны с использованием компьютера. Сфера применения компьютера постоянно расширяется, существенно влияя на развитие производительных сил нашего общества. Непрерывно изменяются технико-экономические характеристики компьютера, например, такие, как быстрота действия, ёмкость памяти, надёжность в работе, стоимость, удобства в эксплуатации, габаритные размеры, потребляемая мощность и др. В широком понимании всякий компьютер рассматривается как преобразователь информации.

Целью данного курсового проекта является создание приложения для работы с файлами в ОС Windows на языке C#.

При разработке приложения, необходимо решить следующие задачи:

* вывод дисков, каталогов файловой системы на экран;
* просмотр файлов;
* копирование файлов;
* создание каталогов;
* удаление файлов;
* переименование файлов.

# 1 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

## 1.1 Язык программирования C#

«C**#»** (произносится *си шарп*) — [объектно-ориентированный](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) [язык программирования](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F). Разработан в [1998](https://ru.wikipedia.org/wiki/1998)—[2001 годах](https://ru.wikipedia.org/wiki/2001_%D0%B3%D0%BE%D0%B4) группой инженеров компании [Microsoft](https://ru.wikipedia.org/wiki/Microsoft) под руководством [Андерса Хейлсберга](https://ru.wikipedia.org/wiki/%D0%A5%D0%B5%D0%B9%D0%BB%D1%81%D0%B1%D0%B5%D1%80%D0%B3,_%D0%90%D0%BD%D0%B4%D0%B5%D1%80%D1%81) и Скотта Вильтаумота как язык разработки приложений для платформы «[Microsoft .NET Framework](https://ru.wikipedia.org/wiki/.NET_Framework)» и «[.NET Core](https://ru.wikipedia.org/wiki/.NET_Core)». Впоследствии был стандартизирован как [ECMA](https://ru.wikipedia.org/wiki/ECMA)-334 и [ISO](https://ru.wikipedia.org/wiki/ISO)/[IEC](https://ru.wikipedia.org/wiki/IEC) 23270.

«C#» относится к семье языков с [C-подобным синтаксисом](https://ru.wikipedia.org/wiki/C-%D0%BF%D0%BE%D0%B4%D0%BE%D0%B1%D0%BD%D1%8B%D0%B9_%D1%81%D0%B8%D0%BD%D1%82%D0%B0%D0%BA%D1%81%D0%B8%D1%81), из них его синтаксис наиболее близок к «[C++](https://ru.wikipedia.org/wiki/C%2B%2B)» и «[Java](https://ru.wikipedia.org/wiki/Java)». Язык имеет [статическую типизацию](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%82%D0%B8%D0%BF%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F), поддерживает [полиморфизм](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%B8%D0%BC%D0%BE%D1%80%D1%84%D0%B8%D0%B7%D0%BC_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)), [перегрузку операторов](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%B3%D1%80%D1%83%D0%B7%D0%BA%D0%B0_%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BE%D0%B2) (в том числе операторов явного и неявного приведения типа), [делегаты](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%BB%D0%B5%D0%B3%D0%B0%D1%82_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), атрибуты, [события](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D0%B1%D1%8B%D1%82%D0%B8%D0%B9%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), [переменные](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), [свойства](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%BE%D0%B9%D1%81%D1%82%D0%B2%D0%BE_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), [обобщённые](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D0%BE%D0%B1%D1%89%D1%91%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) типы и методы, [итераторы](https://ru.wikipedia.org/wiki/%D0%98%D1%82%D0%B5%D1%80%D0%B0%D1%82%D0%BE%D1%80), [анонимные функции](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%BE%D0%BD%D0%B8%D0%BC%D0%BD%D0%B0%D1%8F_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F) с поддержкой [замыканий](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%BC%D1%8B%D0%BA%D0%B0%D0%BD%D0%B8%D0%B5_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), [LINQ](https://ru.wikipedia.org/wiki/Language_Integrated_Query), [исключения](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%D0%B8%D1%81%D0%BA%D0%BB%D1%8E%D1%87%D0%B5%D0%BD%D0%B8%D0%B9), [комментарии](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BC%D0%B5%D0%BD%D1%82%D0%B0%D1%80%D0%B8%D0%B8_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) в формате «[XML](https://ru.wikipedia.org/wiki/XML)».

Переняв многое от своих предшественников — языков «[C++](https://ru.wikipedia.org/wiki/C%2B%2B)», «[Delphi](https://ru.wikipedia.org/wiki/Delphi_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F))», [Модула](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D1%83%D0%BB%D0%B0-2), «[Smalltalk](https://ru.wikipedia.org/wiki/Smalltalk" \o "Smalltalk)» и, в особенности, «[Java](https://ru.wikipedia.org/wiki/Java)» — «С#», опираясь на практику их использования, исключает некоторые модели, зарекомендовавшие себя как проблематичные при разработке программных систем, например, «C#» в отличие от «C++» не поддерживает [множественное наследование](https://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BD%D0%B0%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) классов (между тем допускается [множественная реализация интерфейсов](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_(%D0%BE%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)#%D0%9C%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BD%D0%B0%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%B8_%D1%80%D0%B5%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81%D0%BE%D0%B2)).

«С#»‎ разрабатывался как язык программирования прикладного уровня для «[CLR](https://ru.wikipedia.org/wiki/Common_Language_Runtime)» и, как таковой, зависит, прежде всего, от возможностей самой «CLR». Это касается, прежде всего, системы типов «С#»‎, которая отражает «[BCL](https://ru.wikipedia.org/wiki/Base_Class_Library)». Присутствие или отсутствие тех или иных выразительных особенностей языка диктуется тем, может ли конкретная языковая особенность быть транслирована в соответствующие конструкции «CLR». Так, с развитием «CLR» от версии 1.1 к 2.0 значительно обогатился и сам «C#»; подобного взаимодействия следует ожидать и в дальнейшем (однако, эта закономерность была нарушена с выходом «C# 3.0», представляющего собой расширения языка, не опирающиеся на расширения платформы .NET). «CLR» предоставляет «С#»‎, как и всем другим [.NET](https://ru.wikipedia.org/wiki/.NET_Framework)-ориентированным языкам, многие возможности, которых лишены «классические» языки программирования. Например, [сборка мусора](https://ru.wikipedia.org/wiki/%D0%A1%D0%B1%D0%BE%D1%80%D0%BA%D0%B0_%D0%BC%D1%83%D1%81%D0%BE%D1%80%D0%B0) не реализована в самом «C#»‎, а производится «CLR» для программ, написанных на «C#» точно так же, как это делается для программ на «[VB.NET](https://ru.wikipedia.org/wiki/Visual_Basic_.NET)», «[J#](https://ru.wikipedia.org/w/index.php?title=Visual_J&action=edit&redlink=1)» и др.

## 1.2 Устройство файловой системы ОС «Windows»

Ядром операционной системы является модуль, который обеспечивает управление файлами — файловая система.

Основная задача файловой системы — обеспечение взаимодействия программ и физических устройств ввода/вывода (различных накопителей). Она также определяет структуру хранения файлов и каталогов на диске, правила задания имен файлов, допустимые атрибуты файлов, права доступа и др.

Обычно файловую систему воспринимают и как средство управления файлами, и как общее хранилище файлов.

Файл — это поименованная последовательность любых данных, стандартная структура которой обеспечивает ее размещение в памяти машины. Файл может содержать программу, числовые данные, текст, закодированное изображение или звук и др. Для каждого файла на диске выделяется поименованная область, причем файл не требует для своего размещения непрерывное пространство, так как может занимать свободные кластеры в разных частях диска.

Имя файла — это символьная строка, правила построения которой зависят от конкретной файловой системы. Максимальная длина имени файла в Windows составляет 255 символов. Имена могут содержать любые символы, включая пробелы, кроме следующих: прямой и обратный слэш (\ и /), двоеточие (:), звездочка (\*), знак вопроса (?), двойная кавычка ("), знаков меньше и больше (< и >), знака «трубопровода» (|). Система сохраняет использованные в длинных именах строчные буквы.

Помимо имени, файл имеет расширение (тип) длиной до 3 символов, которое отделяется от имени точкой. К свойствам файла также относятся: реальный размер и объем занимаемого дискового пространства; время создания, последнего изменения и доступа; имя создателя файла; пароль для доступа, атрибуты и др.

Файл может иметь следующие атрибуты:

* R (Read-Only) — «только для чтения». При попытке модифицировать или удалить файл с этим атрибутом будет выдано соответствующее сообщение.
* H (Hidden) — «скрытый файл». При просмотре содержимого папки (без специальных установок или ключа) сведения о файлах с таким атрибутом не выдаются.
* A (Archive) — «неархивированный файл». Этот атрибут устанавливается при создании каждого файла и снимается средствами архивации и резервирования файлов.

Для удобства работы с файлами и их систематизации на диске создаются папки (каталоги), структура которых определяет логическую организацию данных.

Папка (каталог) — это специальное место на диске, в котором хранятся имена файлов, сведения об их размерах, времени последнего обновления и т.д. Имена папок образуются по тем же правилам, что и имена файлов.

Структура папок в «Windows» иерархическая (древовидная). Папка самого верхнего уровня — главная (корневая) — создается автоматически и не имеет имени. В ней находятся сведения не только файлов, но и о папках первого уровня (папки первого и последующих уровней создаются пользователем). Папка, с которой в данный момент работает пользователь, называется текущей.

С папками и файлами могут выполняться операции создания, удаления, копирования и перемещения, а также изменение их свойств и управление доступом.

Физическая организация данных на носителе зависит от файловой системы, которая предусматривает выделение в процессе форматирования диска специальных областей: системной области и области данных. Основными компонентами системной области являются: загрузочная запись, таблицы размещения файлов и корневой каталог (папка). Область данных содержит файлы и папки.

Вся область данных диска делится на кластеры, которые представляют собой неделимые блоки данных одного размера на диске. Все кластеры пронумерованы. В самом начале диска размещается таблица размещения файлов, содержащая столько записей, сколько кластеров доступно на диске. В ней содержатся сведения о номерах кластеров, в которых размещается файл, отмечены неиспользуемые кластеры, а также поврежденные кластеры, которые помечаются определенным значением, после чего уже никогда не употребляются.

Каждый кластер файла содержит номер следующего в цепочке его кластеров. Таким образом, достаточно знать номер первого кластера в цепочке, который хранится в оглавлении диска, чтобы определить номера всех кластеров, содержащих данный файл. Занимаемый файлом объем кратен количеству кластеров. Наличие у каждого кластера индивидуального номера позволяет найти область расположения файла, причем необязательно, чтобы его кластеры располагались рядом. Если разные фрагменты файла располагаются в несмежных кластерах, то говорят о фрагментации файла.

Каждый диск на компьютере имеет уникальное имя. Диски именуются буквами латинского алфавита. Обычно накопителю на гибком магнитном диске (НГМД) присваивается имя А:, а винчестеру (НЖМД) — С:.

Жесткий диск представляет собой физическое устройство. Для организации эффективной работы с дисковым пространством жесткого магнитного диска с помощью специальной программы его разбивают на ряд разделов — логических дисков, каждый из которых рассматривается системой как отдельный диск и именуется последующими буквами латинского алфавита (D, E и т.д.).

«Windows XP» позволяет форматировать жесткий диск в файловой системе «FAT» или «NTFS».

Система «FAT (File Allocation Table)» — представляет собой таблицу размещения файлов «MS-DOS» и «Windows 9x», поэтому понимается этими ОС. Но она имеет низкую отказоустойчивость, и при аварийном отключении питания велика вероятность потери данных.

Система «NTFS (New Technology File System)» — была разработана Microsoft специально для «Windows NT». Она гарантирует сохранность данных в случае копирования даже при программно-аппаратном сбое или отключении электропитания, превосходит «FAT» по эффективности использования ресурсов (например, работает с файлами размером более 4 Гб), предоставляет возможность создавать «динамические» жесткие диски, объединяющие несколько папок, предоставляет средства для разграничения доступа и защиты информации и др.

Перевод логического диска из «FAT» в «NTFS» осуществляется штатной программой Windows или специальными программами без потери информации. Также существуют специальные программы, которые могут производить конвертацию из «NTFS» в «FAT», однако в большинстве случаев такой перевод требует форматирования диска.

## 1.3 Работа с файловой системой в языке программирования C#

### 1.3.1 Работа с дисками

Фреймворк .NET предоставляет большие возможности по управлению и манипуляции файлами и каталогами, которые по большей части сосредоточены в пространстве имен «System.IO». Классы, расположенные в этом пространстве имен (такие как «Stream», «StreamWriter», «FileStream» и др.), позволяют управлять файловым вводом-выводом.

Работу с файловой системой начинается с самого верхнего уровня - дисков. Для представления диска в пространстве имен «System.IO» имеется класс «DriveInfo».

Этот класс имеет статический метод «GetDrives()», который возвращает имена всех логических дисков компьютера. Также он предоставляет ряд полезных свойств:

* «AvailableFreeSpace», указывает на объем доступного свободного места на диске в байтах;
* «DriveFormat», получает имя файловой системы;
* «DriveType», представляет тип диска;
* «IsReady», готов ли диск (например, DVD-диск может быть не вставлен в дисковод);
* «Name», получает имя диска;
* «RootDirectory», возвращает корневой каталог диска;
* «TotalFreeSpace», получает общий объем свободного места на диске в байтах;
* «TotalSize», общий размер диска в байтах;
* «VolumeLabel», получает или устанавливает метку тома.

### 1.3.2 Работа с каталогами

Для работы с каталогами в пространстве имен System.IO предназначены сразу два класса: «Directory» и «DirectoryInfo»

Статический класс Directory предоставляет ряд методов для управления каталогами. Некоторые из этих методов:

* «CreateDirectory(path)», создает каталог по указанному пути;
* «Delete(path)», удаляет каталог по указанному пути;
* «Exists(path)»: определяет, существует ли каталог по указанному пути;
* «GetCurrentDirectory()», получает путь к текущей папке;
* «GetDirectories(path)», получает список подкаталогов в каталоге;
* «GetFiles(path)», получает список файлов в каталоге;
* «Move(sourceDirName, destDirName)», перемещает каталог;
* «GetParent(path)», получение родительского каталога;
* «GetLastWriteTime(path)», возвращает время последнего изменения каталога.

Класс «DirectoryInfo» предоставляет функциональность для создания, удаления, перемещения и других операций с каталогами. Во многом он похож на «Directory», но не является статическим.

Основные методы класса «DirectoryInfo»:

* «Create()», создает каталог;
* «CreateSubdirectory(path)», создает подкаталог по указанному пути;
* «Delete()», удаляет каталог;
* «GetDirectories()», получает список подкаталогов папки в виде массива;
* «GetFiles()», получает список файлов в папке в виде массива;
* «MoveTo(destDirName)», перемещает каталог.

Основные свойства класса «DirectoryInfo»:

* «CreationTime», представляет время создания каталога;
* «LastAccessTime», представляет время последнего доступа к каталогу;
* «LastWriteTime», представляет время последнего изменения каталога;
* «Exists», определяет, существует ли каталог;
* «Parent», получение родительского каталога;
* «Root», получение корневого каталога;
* «Name», имя каталога;
* «FullName», полный путь к каталогу.

Как видно из функционала, оба класса предоставляют похожие возможности. Когда же и что использовать? Если надо совершить одну-две операции с одним каталогом, то проще использовать класс «Directory». Если необходимо выполнить последовательность операций с одним и тем же каталогом, то лучше воспользоваться классом «DirectoryInfo». Методы класса «Directory» выполняют дополнительные проверки безопасности. А для класса «DirectoryInfo» такие проверки не всегда обязательны.

### 1.3.3 Работа с файлами

Подобно паре «Directory/DirectoryInfo» для работы с файлами предназначена пара классов «File» и «FileInfo». С их помощью мы можем создавать, удалять, перемещать файлы, получать их свойства и многое другое.

Некоторые полезные методы и свойства класса «FileInfo»:

* «CopyTo(path)», копирует файл в новое место по указанному пути;
* «Create()», создает файл;
* «Delete()», удаляет файл;
* «MoveTo(destFileName)», перемещает файл в новое место;
* Свойство «Directory», получает родительский каталог в виде объекта;
* Свойство «DirectoryName», получает полный путь к родительскому каталогу;
* Свойство «Exists», указывает, существует ли файл;
* Свойство «Length», получает размер файла;
* Свойство «Extension», получает расширение файла;
* Свойство «Name», получает имя файла;
* Свойство «FullName», получает полное имя файла.

Класс File реализует похожую функциональность с помощью статических методов:

* «Copy()», копирует файл в новое место;
* «Create()», создает файл;
* «Delete()», удаляет файл;
* «Move», перемещает файл в новое место;
* «Exists(file)», определяет, существует ли файл.

# 2 ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ ПРОЕКТА

## 2.1 Формирование структуры проекта

Разработка алгоритма функционирования программного продукта проводилась нисходящим методом, который заключается в том, что разрабатываемый алгоритм разделяется на дочерние алгоритмы, а те, в свою очередь, разделяются на элементарные алгоритмы. Структуру проекта можно представить в виде нескольких блоков. В первом блоке выполняется запуск программы, в ходе которого происходит инициализация главного окна приложения. В втором блоке осуществляется просмотр и редактирование данных (добавление, изменение, удаление). В третьем блоке осуществляется вывод данных на экран.

## 2.2 Алгоритмическое обеспечение проекта

В ходе проектирования программного продукта исследована предметная область, собрана необходимая информация, определены цели и задачи проектирования базы данных. Необходимо выполнить концептуальное, логическое и физическое проектирование приложения. Интерфейс программы должен быть простым и удобным для пользователя.

## 2.3 Реализация программного продукта

Процесс создания программного продукта делится на несколько стадий:

* изучение выбранной предметной области;
* обработка полученной информации;
* выбор средств реализации;
* программирование;
* тестирование;
* эксплуатация.

Наиболее целесообразный метод тестирования – проверка работоспособности основных функций программы. Программа должна нормально функционировать при внесении информации и т. п. Процесс создания программного продукта делится на несколько стадий: изучение выбранной предметной области; обработка полученной информации; выбор средств реализации; программирование; тестирование; эксплуатация.

Для запуска приложения, необходимо иметь на компьютере «.NET Framework». После приложения мы видим главное окно программы, изображенное на рисунке 2.1.

На главном окне приложения отображаются две области содержащие диски, каталоги компьютера. Так же внизу окна присутствует подсказка по командам приложения.

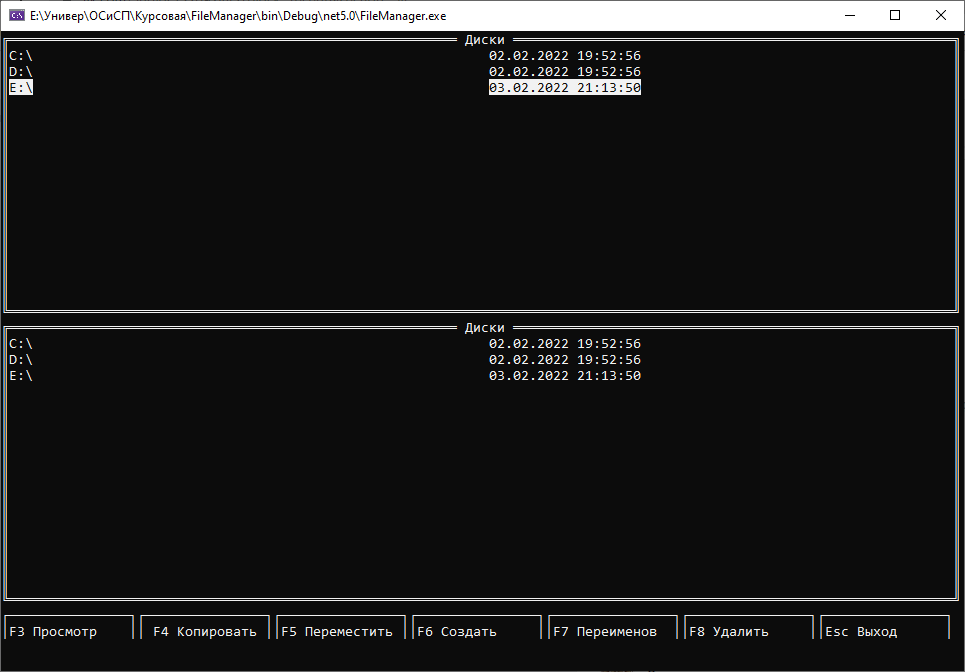


Рисунок 2.1 – Главное окно приложения

В приложение есть возможность перемещаться между дисками и каталоги. Так же по нажатию клавиши Tab можно переключаться между окнами приложения. Пример отображения каталогов, файлов на диске представлен на рисунке 2.2.

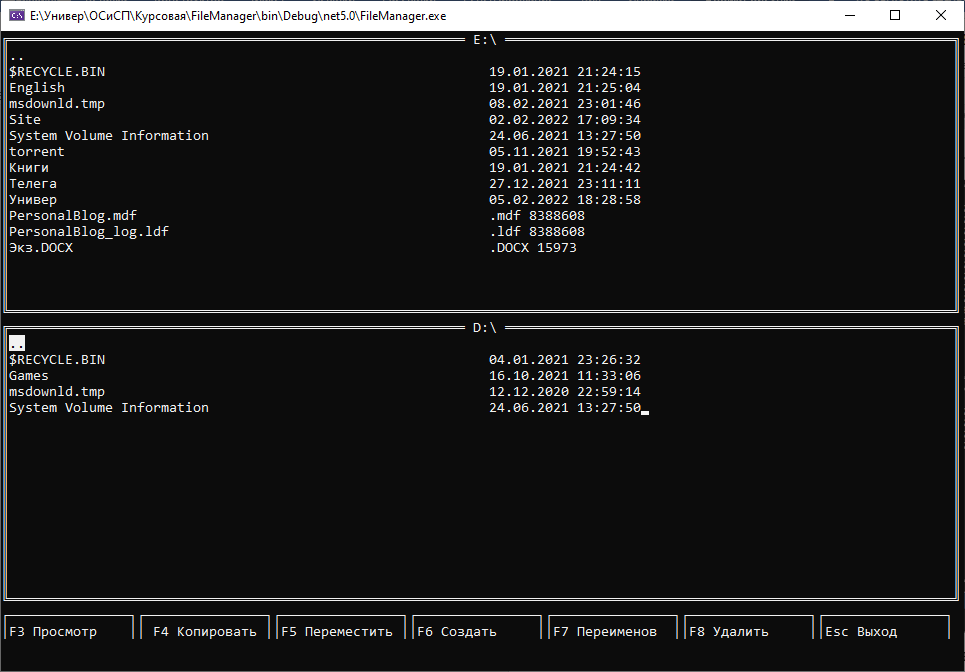


Рисунок 2.2 – Пример отображения каталогов и файлов

Команда «Просмотр» позволяет просмотреть содержимое файла в текстовом виде. Пример отображения содержимого файла представлен на рисунке 2.3.

Команда «Копировать», копирует файл в ту директорию, которая выбрана в втором окне. Пример копирования файла «Экз.DOCX» с диска Е на диск D показан на рисунке 2.4.

Команда «Переместить» позволяет переместить файл с из одной директории в другую. Пример перемещения файла показан на рисунке 2.5.

Команда «Создать» позволяет создать каталог с указанным именем на выбранном диске. Пример создания каталога с именем «Тестовый каталог» представлен на рисунке 2.6.

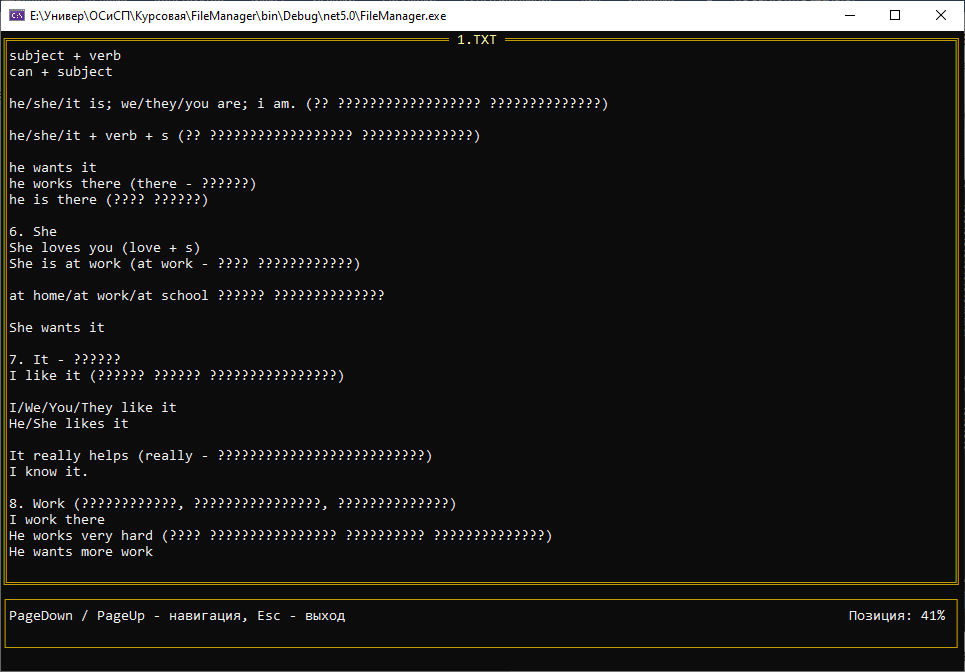


Рисунок 2.3 – Пример отображения содержимого файла

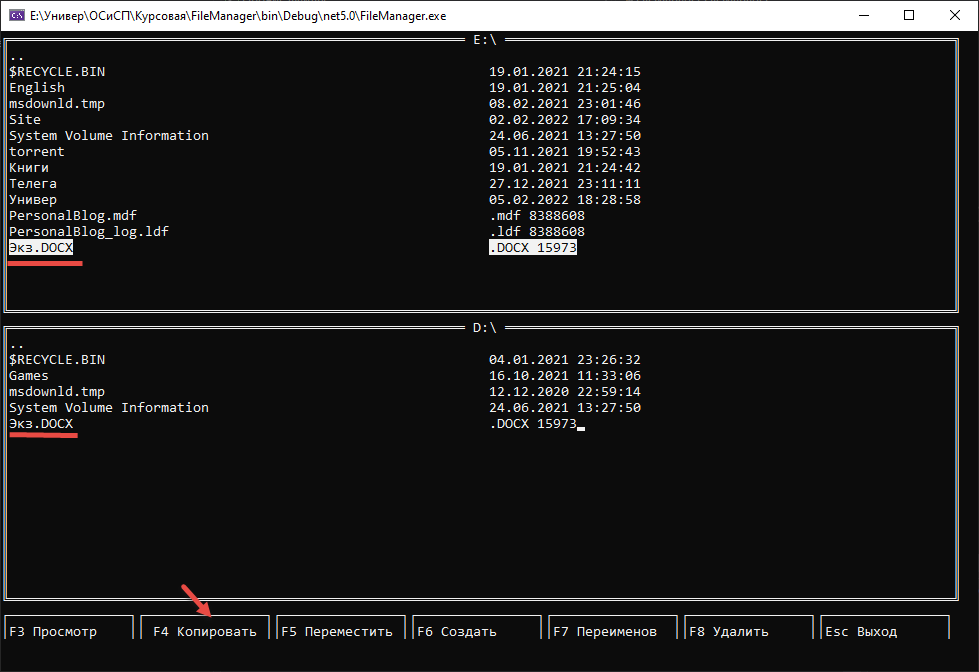


Рисунок 2.4 – Пример копирования файла

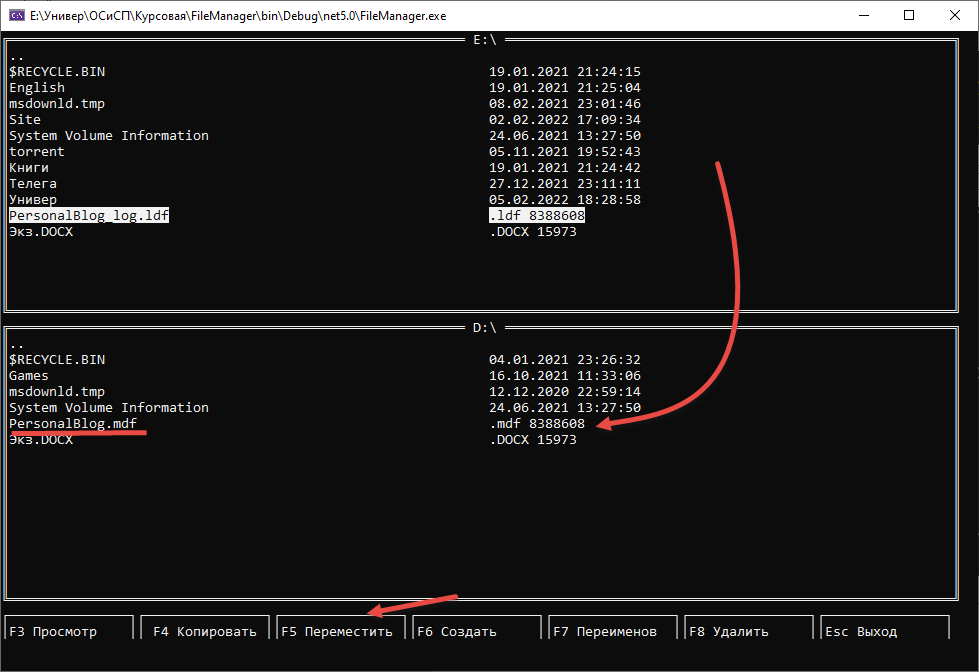


Рисунок 2.5 – Пример перемещения файла

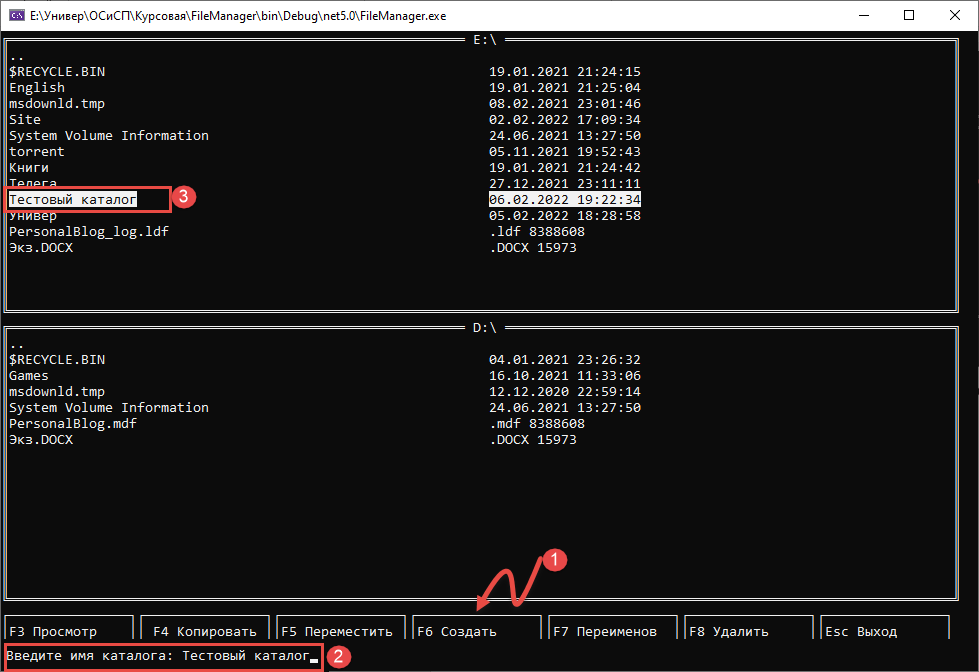


Рисунок 2.6 – Пример создания каталога

Команда «Переименовать» позволяет переименовать каталог или файл. Пример переименования каталога показан на рисунке 2.7.

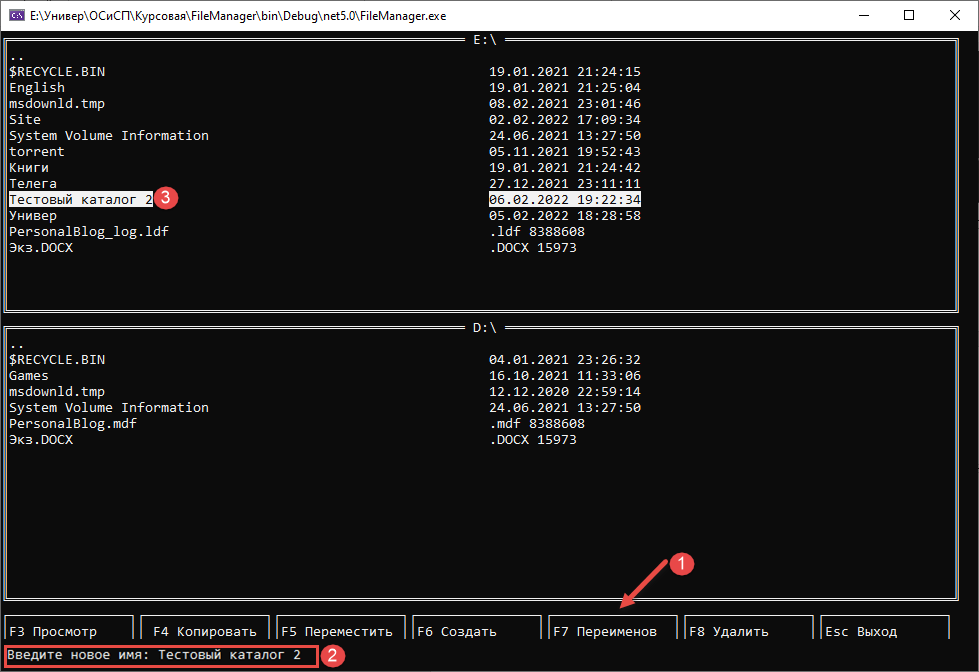


Рисунок 2.7 – Пример переименование каталога

Команда «Удалить» позволяет удалить каталог, файл.

На этом реализация программного продукта закончена.

Листинги кода основных классов, представлены в приложении А.

# ЗАКЛЮЧЕНИЕ

В ходе курсовой работы была спроектировано и разработано приложения для работы с файловой системы ОС «Windows». Приложение было реализовано на языке программирования «C#». Программа предоставляет возможность ввод, удаление, хранения, редактирования и просмотр информации. Реализованное приложение будет полезна людям, для работы с файлами, каталогами. По окончанию разработки программного продукта были достигнуты следующие цели:

* вывод дисков, каталогов файловой системы на экран;
* просмотр файлов;
* копирование файлов;
* создание каталогов;
* удаление файлов;
* переименование файлов.

Разработка программы позволила мне укрепить знания в работе с файловой системой ОС «Windows».

Таким образом, цель курсового проектирования была успешно достигнута.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. «Руководство по C#» [Электронный ресурс] – Режим доступа - https://metanit.com/sharp/tutorial/ – Дата доступа: 02.02.2022.
2. «Примеры работы с файловой системой в C#» [Электронный ресурс] – Режим доступа - https://metanit.com/sharp/tutorial/5.1.php/ – Дата доступа: 02.02.2022.
3. «Устройство файловой системы в ОС Windows» [Электронный ресурс] – Режим доступа - https://studopedia.ru/3\_27272\_faylovaya-sistema-Windows.html/ – Дата доступа: 02.02.2022.
4. «.NET Framework документация» [Электронный ресурс] – Режим доступа - https://docs.microsoft.com/ru-ru/dotnet/framework/ – Дата доступа: 29.01.2022.
5. «Документация C#» [Электронный ресурс] – Режим доступа: https://docs.microsoft.com/ru-ru/dotnet/csharp/ – Дата доступа: 25.01.2022.

# ПРИЛОЖЕНИЕ А

Листинг класса «Explorer», главный класс управляющий приложением и реализующий методы работы с файловой системой:

class Explorer

{

public static int HEIGHT\_KEYS = 3;

public static int BOTTOM\_OFFSET = 2;

public event OnKey KeyPress;

List<FilePanel> panels = new List<FilePanel>();

private int activePanelIndex;

static Explorer()

{

Console.SetWindowSize(120, 40);

Console.SetBufferSize(120, 40);

}

public Explorer()

{

FilePanel filePanel = new FilePanel();

filePanel.Top = 0;

filePanel.Left = 0;

this.panels.Add(filePanel);

filePanel = new FilePanel();

filePanel.Top = FilePanel.PANEL\_HEIGHT;

filePanel.Left = 0;

this.panels.Add(filePanel);

activePanelIndex = 0;

this.panels[this.activePanelIndex].Active = true;

KeyPress += this.panels[this.activePanelIndex].KeyboardProcessing;

foreach (FilePanel fp in panels)

{

fp.Show();

}

this.ShowKeys();

}

public void Run()

{

bool exit = false;

while (!exit)

{

if (Console.KeyAvailable)

{

this.ClearMessage();

ConsoleKeyInfo userKey = Console.ReadKey(true);

switch (userKey.Key)

{

case ConsoleKey.Tab:

this.ChangeActivePanel();

break;

case ConsoleKey.Enter:

this.ChangeDirectoryOrRunProcess();

break;

case ConsoleKey.F3:

this.ViewFile();

break;

case ConsoleKey.F4:

this.Copy();

break;

case ConsoleKey.F5:

this.Move();

break;

case ConsoleKey.F6:

this.CreateDirectory();

break;

case ConsoleKey.F7:

this.Rename();

break;

case ConsoleKey.F8:

this.Delete();

break;

case ConsoleKey.Escape:

exit = true;

Console.ResetColor();

Console.Clear();

break;

case ConsoleKey.DownArrow:

goto case ConsoleKey.PageUp;

case ConsoleKey.UpArrow:

goto case ConsoleKey.PageUp;

case ConsoleKey.End:

goto case ConsoleKey.PageUp;

case ConsoleKey.Home:

goto case ConsoleKey.PageUp;

case ConsoleKey.PageDown:

goto case ConsoleKey.PageUp;

case ConsoleKey.PageUp:

this.KeyPress(userKey);

break;

default:

break;}}}}

private void ChangeDirectoryOrRunProcess()

{

FileSystemInfo fsInfo = this.panels[this.activePanelIndex].GetActiveObject();

if (fsInfo != null)

{

if (fsInfo is DirectoryInfo)

{

try

{

Directory.GetDirectories(fsInfo.FullName);

}

catch

{

return;

}

this.panels[this.activePanelIndex].Path = fsInfo.FullName;

this.panels[this.activePanelIndex].SetLists();

this.panels[this.activePanelIndex].UpdatePanel();

}

else

{

Process.Start(((FileInfo)fsInfo).FullName);

}

}

else

{

string currentPath = this.panels[this.activePanelIndex].Path;

DirectoryInfo currentDirectory = new DirectoryInfo(currentPath);

DirectoryInfo upLevelDirectory = currentDirectory.Parent;

if (upLevelDirectory != null)

{

this.panels[this.activePanelIndex].Path = upLevelDirectory.FullName;

this.panels[this.activePanelIndex].SetLists();

this.panels[this.activePanelIndex].UpdatePanel();

}

else

{

this.panels[this.activePanelIndex].SetDiscs();

this.panels[this.activePanelIndex].UpdatePanel();}}}

private void Copy()

{

foreach (FilePanel panel in panels)

{

if (panel.isDiscs)

{

return;

}

}

if (this.panels[0].Path == this.panels[1].Path)

{

return;

}

try

{

string destPath = this.activePanelIndex == 0 ? this.panels[1].Path : this.panels[0].Path;

FileSystemInfo fileObject = this.panels[this.activePanelIndex].GetActiveObject();

FileInfo currentFile = fileObject as FileInfo;

if (currentFile != null)

{

string fileName = currentFile.Name;

string destName = Path.Combine(destPath, fileName);

File.Copy(currentFile.FullName, destName, true);

}

else

{

string currentDir = ((DirectoryInfo)fileObject).FullName;

string destDir = Path.Combine(destPath, ((DirectoryInfo)fileObject).Name);

CopyDirectory(currentDir, destDir);

}

this.RefreshPannels();

}

catch (Exception e)

{

this.ShowMessage(e.Message);

return;}}

private void CopyDirectory(string sourceDirName, string destDirName)

{

DirectoryInfo dir = new DirectoryInfo(sourceDirName);

DirectoryInfo[] dirs = dir.GetDirectories();

if (!Directory.Exists(destDirName))

{

Directory.CreateDirectory(destDirName);

}

FileInfo[] files = dir.GetFiles();

foreach (FileInfo file in files)

{

string temppath = Path.Combine(destDirName, file.Name);

file.CopyTo(temppath, true);

}

foreach (DirectoryInfo subdir in dirs)

{

string temppath = Path.Combine(destDirName, subdir.Name);

CopyDirectory(subdir.FullName, temppath);

}

}

private void CreateDirectory()

{

if (this.panels[this.activePanelIndex].isDiscs)

{

return;

}

string destPath = this.panels[this.activePanelIndex].Path;

string dirName = this.AksName("Введите имя каталога: ");

try

{

string dirFullName = Path.Combine(destPath, dirName);

DirectoryInfo dir = new DirectoryInfo(dirFullName);

if (!dir.Exists)

{

dir.Create();

}

else

{

this.ShowMessage("Каталог с таким именем уже существует");

}

this.RefreshPannels();

}

catch (Exception e)

{

this.ShowMessage(e.Message);

}

}

private void Delete()

{

if (this.panels[this.activePanelIndex].isDiscs)

{

return;

}

FileSystemInfo fileObject = this.panels[this.activePanelIndex].GetActiveObject();

try

{

if (fileObject is DirectoryInfo)

{

((DirectoryInfo)fileObject).Delete(true);

}

else

{

((FileInfo)fileObject).Delete();

}

this.RefreshPannels();

}

catch (Exception e)

{

this.ShowMessage(e.Message);

return;

}

}

private void Rename()

{

if (this.panels[this.activePanelIndex].isDiscs)

{

return;

}

FileSystemInfo fileObject = this.panels[this.activePanelIndex].GetActiveObject();

string currentPath = this.panels[this.activePanelIndex].Path;

string newName = this.AksName("Введите новое имя: ");

string newFullName = Path.Combine(currentPath, newName);

try

{

if (fileObject is FileInfo)

{

((FileInfo)fileObject).MoveTo(newFullName);

}

else

{

((DirectoryInfo)fileObject).MoveTo(newFullName);

}

this.RefreshPannels();

}

catch (Exception e)

{

this.ShowMessage(e.Message);

}

}

private void Move()

{

foreach (FilePanel panel in panels)

{

if (panel.isDiscs)

{

return;

}

}

if (this.panels[0].Path == this.panels[1].Path)

{

return;

}

try

{

string destPath = this.activePanelIndex == 0 ? this.panels[1].Path : this.panels[0].Path;

FileSystemInfo fileObject = this.panels[this.activePanelIndex].GetActiveObject();

string objectName = fileObject.Name;

string destName = Path.Combine(destPath, objectName);

if (fileObject is FileInfo)

{

((FileInfo)fileObject).MoveTo(destName);

}

else

{

((DirectoryInfo)fileObject).MoveTo(destName);

}

this.RefreshPannels();

}

catch (Exception e)

{

this.ShowMessage(e.Message);

return;

}

}

private void ViewFile()

{

if (this.panels[this.activePanelIndex].isDiscs)

{

return;

}

FileSystemInfo fileObject = this.panels[this.activePanelIndex].GetActiveObject();

if (fileObject is DirectoryInfo || fileObject == null)

{

return;

}

if (((FileInfo)fileObject).Length == 0)

{

this.ShowMessage("Файл пуст");

return;

}

if (((FileInfo)fileObject).Length > 100000000)

{

this.ShowMessage("Файл слишком большой для просмотра");

return;

}

this.DrawViewFileFrame(fileObject.Name);

string fileContent = this.ReadFileToString(fileObject.FullName, Encoding.ASCII);

int beginPosition = 0;

int symbolCount = 0;

bool endOfFile = false;

bool beginFile = true;

Stack<int> printSymbols = new Stack<int>();

symbolCount = this.PrintStingFrame(fileContent, beginPosition);

printSymbols.Push(symbolCount);

this.PrintProgress(beginPosition + symbolCount, fileContent.Length);

bool exit = false;

while (!exit)

{

endOfFile = (beginPosition + symbolCount) >= fileContent.Length;

beginFile = (beginPosition <= 0);

ConsoleKeyInfo userKey = Console.ReadKey(true);

switch (userKey.Key)

{

case ConsoleKey.Escape:

exit = true;

break;

case ConsoleKey.PageDown:

if (!endOfFile)

{

beginPosition += symbolCount;

symbolCount = this.PrintStingFrame(fileContent, beginPosition);

printSymbols.Push(symbolCount);

this.PrintProgress(beginPosition + symbolCount, fileContent.Length);

}

break;

case ConsoleKey.PageUp:

if (!beginFile)

{

if (printSymbols.Count != 0)

{

beginPosition -= printSymbols.Pop();

if (beginPosition < 0)

{

beginPosition = 0;

}

}

else

{

beginPosition = 0;

}

symbolCount = this.PrintStingFrame(fileContent, beginPosition);

this.PrintProgress(beginPosition + symbolCount, fileContent.Length);

}

break;

}

}

Console.Clear();

foreach (FilePanel fp in panels)

{

fp.Show();

}

this.ShowKeys();

}

# ПРИЛОЖЕНИЕ Б

Содержание электронного носителя

На электронном носителе расположены следующие файлы и директории:

– Файл «Документация.docx» представляет собой текстовый документ с курсовой работой.

– Архив «FileManager.zip» содержит исходный код курсового проекта.