МИНОБРНАУКИ РОССИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Балтийский государственный технический университет «ВОЕНМЕХ» им. Д.Ф. Устинова»
(БГТУ «ВОЕНМЕХ» им. Д.Ф. Устинова)

ОТЧЕТ
о курсовой работе

Тема Объектно-ориентированная разработка программ с графическим пользовательским интерфейсом «сверху-вниз»

Вариант: Архиватор по алгоритму Хаффмана

Обучающегося группы И924Б (группа) Фирсов М.А. (Фамилия и инициалы)

Направление подготовки / специальность 09.03.01 (индекс) Информатика и вычислительная техника (полное наименование направления подготовки / специальности)

Направленность образовательной программы Автоматизированные системы обработки информации и управления (профиль / специализация / магистерская программа)

Дисциплина (модуль) Информационные технологии и программирование

Руководитель: _____ (подпись)

_____ (ученая степень, ученое звание) Вальштейн К.В. (Фамилия ИО)

Оценка: _____

« » _____ 20__ г.

Обучающийся: _____ (подпись)

Фирсов М.А. (Фамилия ИО)

« » _____ 20 23 г.

САНКТ-ПЕТЕРБУРГ
2023 г.

РЕФЕРАТ

Отчет 43 с., 47 рис., 9 источн., 1 прил.

ООП, ГРАФИЧЕСКИЙ ПОЛЬЗОВАТЕЛЬСКИЙ ИНТЕРФЕЙС, НАСЛЕДОВАНИЕ, C#, WPF

Объект разработки – интерактивное приложение с графическим пользовательским интерфейсом.

Целью курсовой работы является разработка интерактивного приложения с графическим пользовательским интерфейсом с использованием фреймворка .Net и технологии WPF.

В процессе работы проводились исследования предметной области, были определены основные требования к разрабатываемому приложению, сформирована иерархия классов, выбраны средства разработки, спроектирован пользовательский интерфейс и структура приложения.

В результате разработки было создано интерактивное приложение с графическим пользовательским интерфейсом «Архиватор по алгоритму Хаффмана».

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 5
1 Постановка задачи 6
2 Описание архиватора 7
2.1 Структура архива *.firsov-archive 7
2.2 Структура упаковщика. Реализация алгоритма Хаффмана 8
2.3 Распаковщик 10
3 Описание иерархии классов 11
3.1 Класс BinaryElement 12
3.2 Класс cName 12
3.3 Класс Code 13
3.4 Класс Dict 14
3.5 Класс Archive 14
3.6 Класс File 15
3.7 Класс FileItem 15
3.8 Класс FileSystemItem 15
3.9 Класс NodeI1O2 16
3.10 Класс NodeI1O3 16
3.11 Класс NodeI1OM 17
3.12 Класс TreeDirectory 18
3.13 Класс ItemDict 19
3.14 Класс PDict 19
3.15 Класс LT 20
3.16 Класс Packer 21
3.17 Класс Unpacker_InStream 23
3.18 Класс Unpacker_OutStream 23
3.19 Класс Unpacker 24
3.20 Класс MainWindow 24
3.21 Класс PackingWindow 26

3.22 Класс UnpackingWindow 27
3.23 Класс CreateFolder 28
3.24 Класс WarningWindow 29
3.25 Демонстрация иерархии классов 29
4 Использованные мультимедийные ресурсы и сторонние библиотеки 31
5 Демонстрация работы 32
ЗАКЛЮЧЕНИЕ 41
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 42
ПРИЛОЖЕНИЕ А Текст программы 43

ВВЕДЕНИЕ

.NET – это фреймворк, который предоставляет обширные возможности для разработки приложений на различных платформах. Он объединяет в себе инструменты, библиотеки и среды выполнения, позволяя использовать различные языки программирования, включая C#, для создания кроссплатформенных решений. .NET обеспечивает высокую производительность, безопасность и удобство разработки.

Windows Presentation Foundation (WPF) является частью .NET и предоставляет инструментарий для разработки графических пользовательских интерфейсов в desktop-приложениях под управлением Windows. Он основан на векторной графике, что обеспечивает масштабируемость интерфейсов на различных устройствах.

Целью курсовой работы является разработка интерактивного приложения с графическим пользовательским интерфейсом с использованием фреймворка .Net и технологии WPF.

Для достижения поставленной цели необходимо решить следующие задачи:

- описать основные требования к разрабатываемому приложению;
- составить иерархию классов;
- разработать приложение;
- продемонстрировать работоспособность приложения.

1 Постановка задачи

Необходимо разработать интерактивное приложение для архиватора, в основе которого бы лежал алгоритм Хаффмана. Сам архиватор должен состоять из двух между собой никак не связанных компонентов, а именно упаковщика и распаковщика.

Упаковщик должен уметь сжимать файлы, если это возможно, и помещать их в архив, который в свою очередь является бинарным файлом. Для удобства к имени архива добавляется расширение файла *.firsov-archive.

Распаковщику необходимо распаковывать как архив целиком, так и выбранную пользователем часть внутри этого архива.

Приложение должно реализовывать файловый менеджер с возможностью перемещаться вниз и вверх по каталогу и архивам *.firsov-archive, а также создавать внутри каталога пустые папки.

За счет интуитивно понятных инструкций графический интерфейс должен позволять пользователю:

- упаковывать и распаковывать файлы;
- создавать пустые папки;
- перемещаться по каталогу и архиву.

В случае неправильных действий пользователя (тех действий, обработка которых не может рассматриваться однозначно или вообще невыполнима), пользователь должен видеть соответствующие предупреждения и причину их вызова.

Взаимодействие с интерфейсом осуществляется при помощи кнопок мыши и клавиатуры, таких как: «вверх», «вниз», «return», «левая кнопка мыши», «скролл вверх» и «скролл вниз».

Программа должна быть написана на C# с использованием WPF, библиотеки описывающие процессы архивации не должны использоваться.

В результате были сформированы требования, согласно которым будет разработано приложение.

2 Описание архиватора

В ходе разработки архиватора была описана модель архива с авторским расширением, с которым могли бы взаимодействовать упаковщик и распаковщик – компоненты архиватора. Без графического интерфейса эти компоненты никак не связаны между собой кроме использования общей концепции файла архива.

2.1 Структура архива *.firsov-archive

Структура архива включает в себя основные элементы:

- имя;
- словарь;
- сжатые данные.

На рисунке 1 схематично показана общая структура архива.

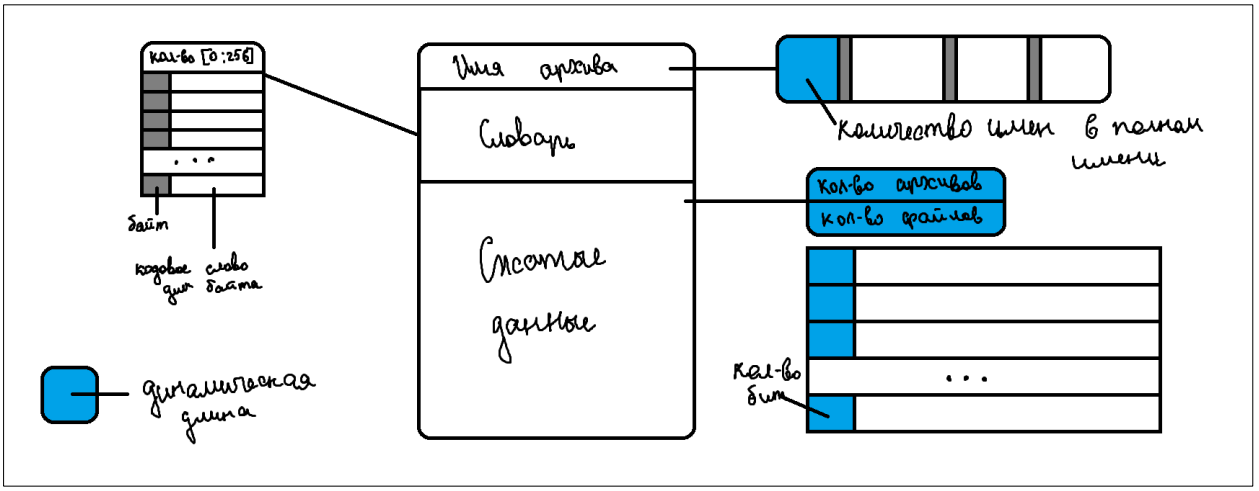

Рисунок 1 – Общая структура архива

Имя состоит из всех имен директории файла в его полном имени. Так длина каждого такого имени не может превышать 255 байт, то можно не записывать символ «\», а вместо него поставить беззнаковый однобайтовый счетчик длины, после которого будет часть имени с заданным количеством байт. Количество таких имен заранее неизвестно, при этом нельзя однозначно установить, счетчика какого размера, гарантированно хватит для хранения информации, поэтому используется динамическая длина. Имена файлов и архивов, за исключением корневого, внутри родительского архива являются локальными по отношению к нему.

Словарь состоит из записей типа «ключ – значение», где ключ – байт, а значение – кодовое слово для кодирования данного байта. Все кодовые слова удовлетворяют условию Фано. Так как ключ может принимать значения от 0 до 256 включительно, за размер словаря должен отвечать двухбайтовый счетчик.

Сжатые данные состоят из префикса – количества архивов и не архивов, и самих битовых последовательностей, содержащих информацию о файлах. Для хранения количества файлов и битов в этих файлах используется динамическая длина.

Если количество неизвестно, используя статическое хранение, определяется максимальный диапазон данных и используется счетчик с минимальным размером, который мы смог уместить этот диапазон. В случае архива, где задача, уменьшить занимаемый объем хранения, данный подход является неоптимальным, поэтому при выборе динамического хранения, как показано на рисунке 2, организация которого похожа на utf-8, можно эффективно использовать ресурсы долговременной памяти.

Рисунок 2 – Организация динмического хранения в памяти

Первые 2 байта служат флагом размера. При чтении сначала читается 1 байт, проверяется флаг, после чего читаются остальные, при записи выбирается минимальный размер, который смог бы уместить количество, после чего добавляется флаг и записываются все байты.

2.2 Структура упаковщика. Реализация алгоритма Хаффмана

После выбора всех необходимых фалов упаковка происходит в 3 этапа:

1) чтение;

2) генерация словаря;

3) запись в архив.

Каждый выбранный файл (кроме архивов, так как они перезаписываются без сжатия) читается, обновляя список счетчиков типа «ключ: значение», где ключ – прочитанный байт, а значение – счетчик встречаемости.

После прочтения всех файлов, список сортируется по ключу счетчик. Далее последовательно, начиная с 1-й вершины, по алгоритму Хаффмана [1] формируется бинарное дерево поиска, где листьями являются элементы списка, а любой отец является суммой своих сыновей. Пример для 6 элементов изображен на рисунке 3.

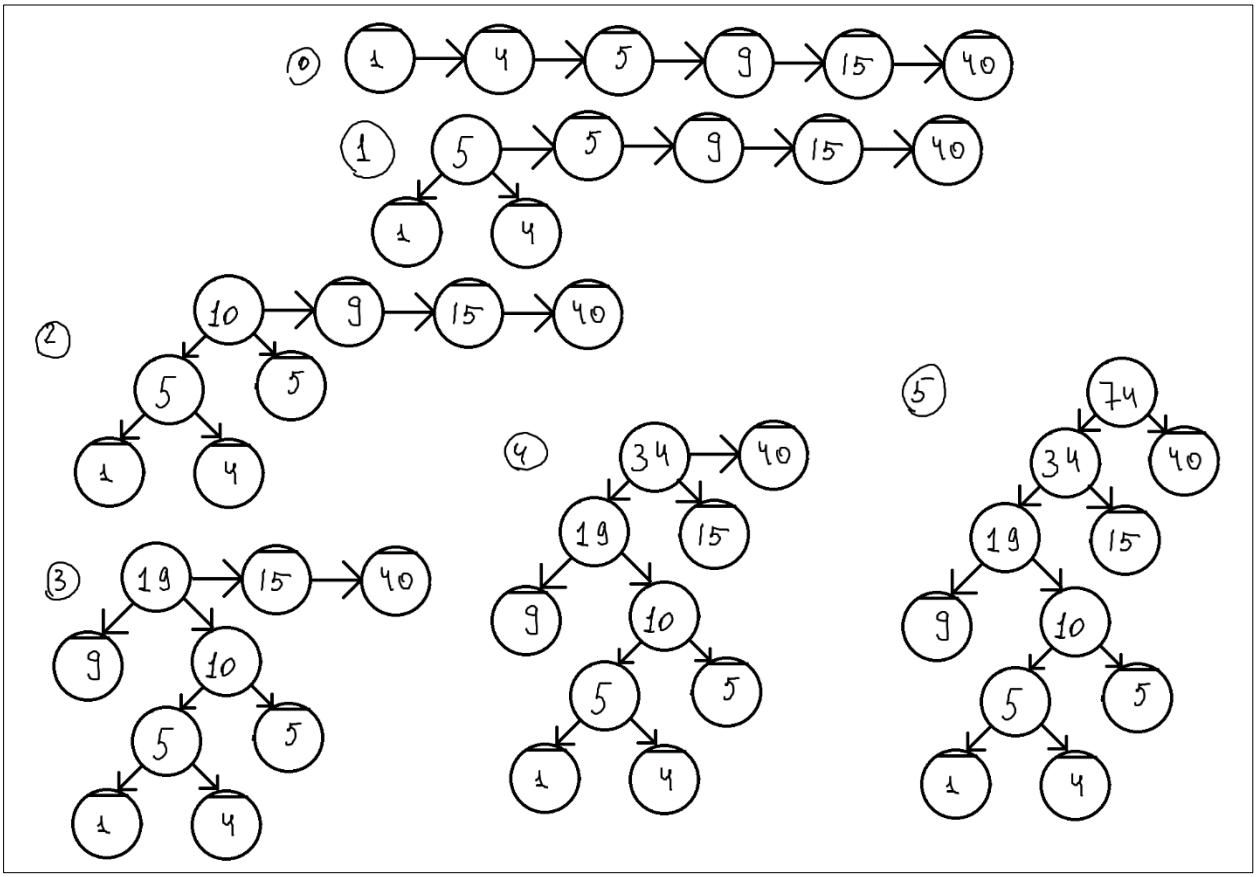

Рисунок 3 – Пример использования алгоритма Хаффмана

Такое дерево удовлетворяет условию Фано, и обеспечивает минимальный размер получаемой последовательности на выходе.

По полученному словарю каждый файл записывается в архив, при этом если записываются вложенные архивы их имена изменяются, всё остальное перезаписывается.

2.3 Распаковщик

Распаковка может быть одного типа из трех возможных:

1) распаковка архива целиком;
2) распаковка 1 файла внутри архива;
3) распаковка нескольких файлов внутри архива.

При это этом распаковка не является рекурсивной, то есть вложенные архивы не будут распакованы, а выбор нескольких файлов возможен только между теми, у которых родительский каталог одинаков, либо он одинаков у папок, где выбрано всё содержимое.

После выбора названий файлов и запуска распаковщика тот читает словарь и генерирует бинарное дерево, листьями которого являются значения байтов. Для каждого имени файла создается реальный в который записываются байты, после каждого спуска по бинарному дереву.

3 Описание иерархии классов

В процессе разработки приложения были созданы следующие классы:

- BinaryElement – абстрактный класс, описывающий интерфейс чтения архива и его составных элементов, как бинарные записи;
- cName – бинарная запись имени файла или архива;
- Code – бинарная запись элемента словаря;
- Dict – бинарная запись словаря;
- Archive – бинарная запись архива;
- File – бинарная запись файла;
- FileItem – представление узла файловой системы;
- FileSystemItem – представление узла непосредственно в списке файлового менеджера;
- NodeI1O2 – узел с 1-м входом и 2-мя выходами;
- NodeI1O3 – узел с 1-м входом и 3-мя выходами;
- NodeI1OM – узел с 1-м входом и множеством выходами;
- TreeDirectory – представление каталога в виде дерева для просмотра архива;
- ItemDict – элемент словаря упаковщика;
- PDict – словарь упаковщика;
- LT – список деревьев;
- Packer – упаковщик (компонент архиватора);
- Unpacker_InStream – входной поток распаковщика;
- Unpacker_OutStream – выходной поток распаковщика;
- Unpacker – распаковщик (компонент архиватора);
- MainWindow – основное окно, в котором отображается файловая система;
- PackingWindow – окно выбора упаковки, открытое в модальном режиме;
- UnpackingWindow – окно выбора распаковки, открытое в

модальном режиме;

- CreateFolder – окно создания папки, открытое в модальном режиме;
- WarningWindow – окно ошибки, открытое в модальном режиме.

3.1 Класс BinaryElement

Класс содержит следующие методы:

- public static ulong _readLength(FileStream fileStream, BinaryReader binaryReader) читает длину, записанную динамически (префикс указывает размер записи) в двоичном коде;
- public void bRead(FileStream fileStream, BinaryReader binaryReader) является абстрактным методом и описывает интерфейс чтения из входного потока составных элементов архива;
- public static bool getBit(byte _byte, byte _index) «вытаскивает» бит по его индексу из байта;
- public static bool getBit(byte[] Data, ulong index) «вытаскивает» бит по его индексу из последовательности байтов.

UML-диаграмма данного класса представлена на рисунке 4.

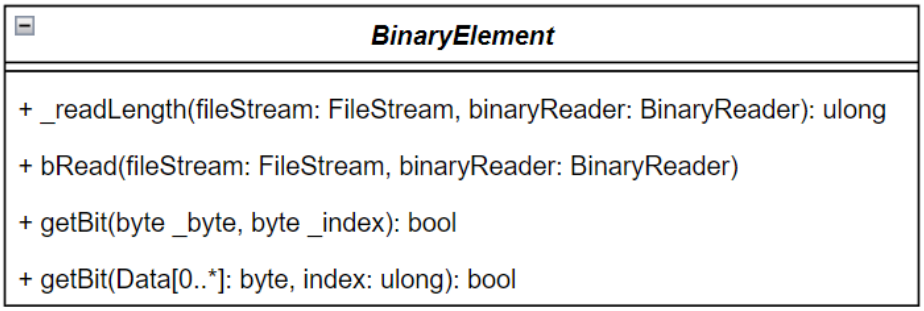

Рисунок 4 – UML-диаграмма класса BinaryElement

3.2 Класс cName

Класс содержит следующие поля:

- public ulong LengthNames – количество локальных имен (имена каждого каталога по пути к файлу, начиная с имени корневой папки, заканчивая названием самого файла);
- public byte[] LengthPartNames – длины каждого локального имени;

- public string[] PartNames – локальные имена.

Класс также включает следующие методы:

- public string toString() возвращает полное название файла;
- public ulong Length() возвращает количество занимаемой в байтах памяти имени в бинарной записи.

UML-диаграмма данного класса представлена на рисунке 5.

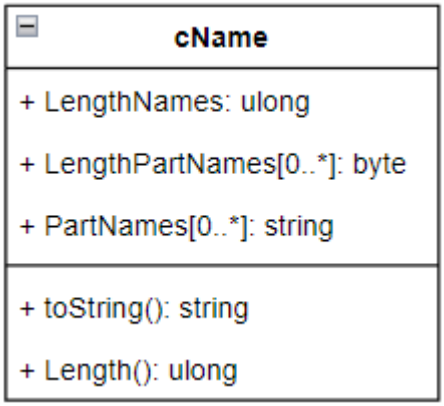

Рисунок 5 – UML-диаграмма класса cName

3.3 Класс Code

Класс содержит следующие поля:

- public byte Key – значение байта;
- public byte LengthBits – количество битов в записи кодового слова;
- public byte LengthBytes – количество байтов в записи кодового слова;
- public byte[] Value – кодовое слово.

UML-диаграмма данного класса представлена на рисунке 6.

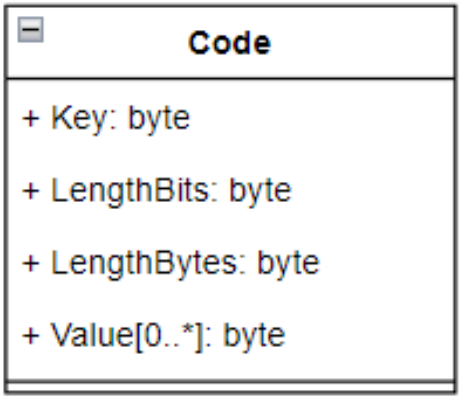

Рисунок 6 – UML-диаграмма класса Code

3.4 Класс Dict

Класс содержит следующие поля:

- public byte Length – количество элементов словаря;
- public Code[] Data – элементы словаря.

UML-диаграмма данного класса представлена на рисунке 7.

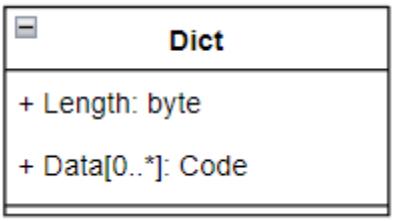

Рисунок 7 – UML-диаграмма класса Dict

3.5 Класс Archive

Класс содержит следующие поля:

- public cName Name – имя архива;
- public ulong Position – позиция в потоке, после имени;
- public ulong LengthArchives – количество дочерних архивов;
- public ulong LengthFiles – количество дочерних файлов;
- public Dict dict – словарь;
- public Archive[] Archives – дочерние архивы;
- public File[] Files – дочерние файлы.

UML-диаграмма данного класса представлена на рисунке 8.

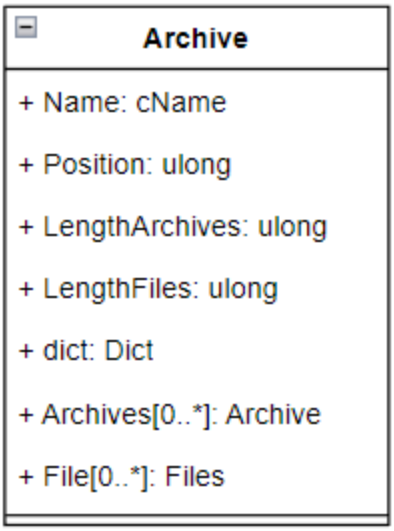

Рисунок 8 – UML-диаграмма класса Archive

3.6 Класс File

Класс содержит следующие поля:

- public cName Name – имя файла;
- public ulong Position – позиция в потоке, после имени;
- public ulong LengthCodeData – длина записи данных в битах.

UML-диаграмма данного класса представлена на рисунке 9.

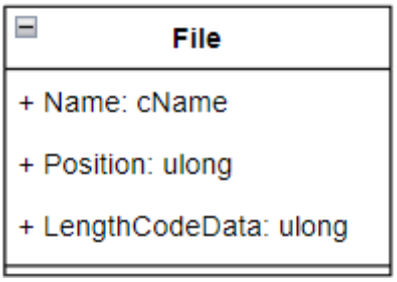

Рисунок 9 – UML-диаграмма класса File

3.7 Класс FileItem

Класс содержит следующие поля:

- public string Name – имя;
- public string Type – тип (архив/файл/каталог);
- public string Size – размер в байтах.

UML-диаграмма данного класса представлена на рисунке 10.

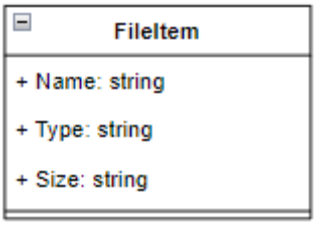

Рисунок 10 – UML-диаграмма класса FileItem

3.8 Класс FileSystemItem

Класс содержит единственное поле public ImageSource imageSource – иконка узла и метод private void chooseImageSource(), который выбирает иконку в соответствии с типом.

UML-диаграмма данного класса представлена на рисунке 11.

Рисунок 11 – UML-диаграмма класса FileSystemItem

3.9 Класс NodeI1O2

Класс содержит следующие поля:

- public byte Value – значение байта (читается если узел – лист);
- public NodeI1O2 left – левый сын;
- public NodeI1O2 right – правый сын.

Класс также включает следующие методы:

- public bool IsLeaf() проверяет является ли текущая вершина листом;
- public byte GetValue() возвращает значение байта;
- public NodeI1O2 SetValue(bool bit_0) спускается по дереву согласно параметру.

UML-диаграмма данного класса представлена на рисунке 12.

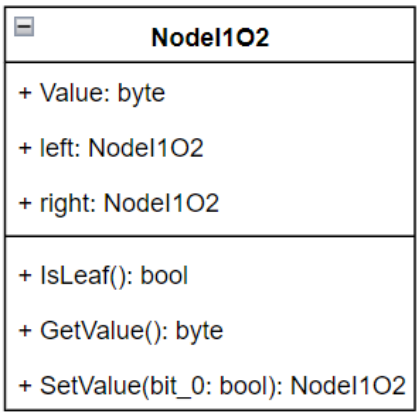

Рисунок 12 – UML-диаграмма класса NodeI1O2

3.10 Класс NodeI1O3

Класс содержит следующие поля:

- public bool FlagRoot – флаг корня;
- public byte Byte – значение байта;
- public ulong Length – количество байтов с заданным значением;
- public NodeI1O3 Left – левый сын в представлении бинарного

дерева;

- public NodeI1O3 Right – правый сын в представлении бинарного дерева;
- public NodeI1O3 Next – следующий элемент в представлении односвязного линейного списка.

Класс также включает единственный метод public void add(), который увеличивает количество на 1.

UML-диаграмма данного класса представлена на рисунке 13.

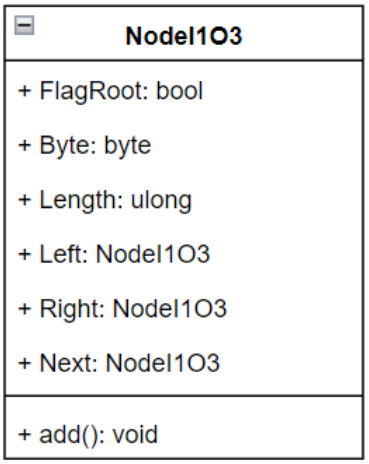

Рисунок 13 – UML-диаграмма класса NodeI1O3

3.11 Класс NodeI1OM

Класс содержит следующие поля:

- public ulong Position – позиция элемента в потоке;
- public string Name – имя элемента;
- public ulong LengthChildren – количество дочерних элементов;
- public LinkedList Child – дочерние элементы;
- public FileItem Info – информация о файле;
- public NodeI1OM Parent – родительский элемент.

Класс также включает следующие методы:

- public NodeI1OM Search(string searchName) ищет элемент с указанным именем среди дочерних;
- public NodeI1OM Add(string currentName) ищет элемент с указанным именем среди дочерних, если не находит, то создает.

UML-диаграмма данного класса представлена на рисунке 14.

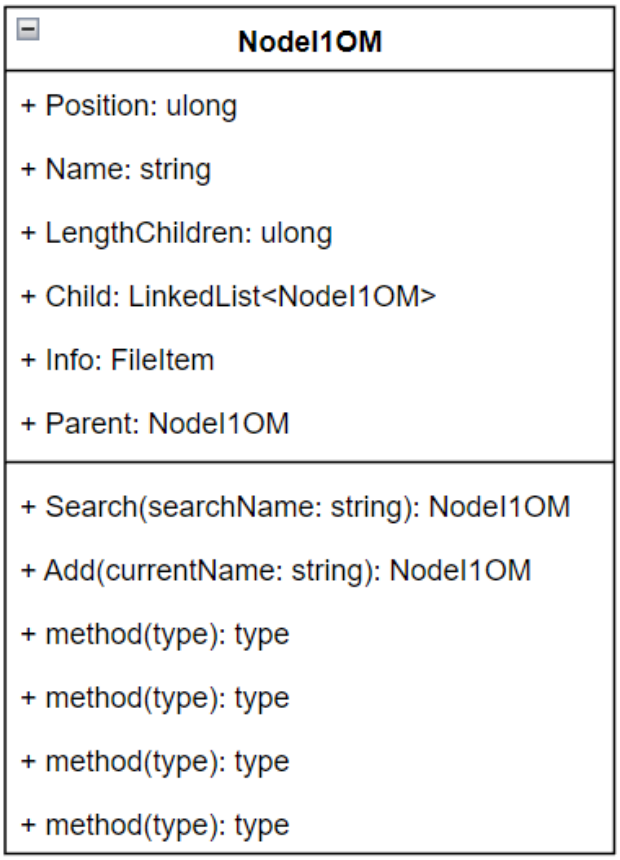

Рисунок 14 – UML-диаграмма класса NodeI1OM

3.12 Класс TreeDirectory

Класс содержит следующие поля:

- public NodeI1OM Root – корень;
- public Archive ArchiveRoot – архив, находящийся в корне.

Класс также включает следующие методы:

- public NodeI1OM Add(string[] PartNames, ulong position) добавляет элемент в дерево;
- public static NodeI1OM Up(NodeI1OM current) позволяет подняться по дереву;
- public static TNodeI1OM Down(NodeI1OM current, string nameChild) позволяет спуститься по дереву.

UML-диаграмма данного класса представлена на рисунке 15.

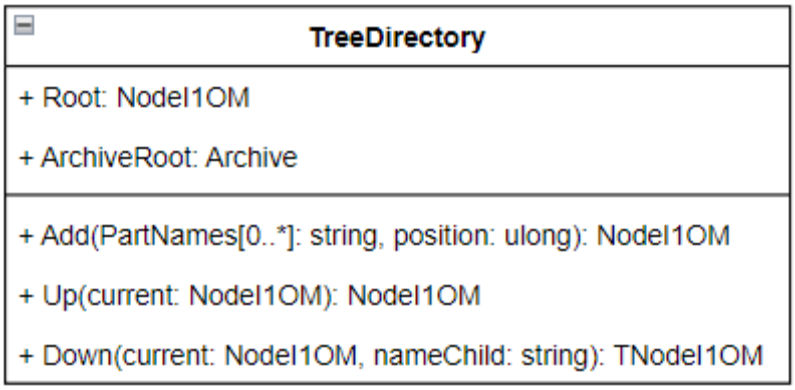

Рисунок 15 – UML-диаграмма класса TreeDirectory

3.13 Класс ItemDict

Класс содержит следующие поля:

- public byte Byte – значение байта;
- public byte Length – длина кодового слова в битах;
- public byte[] Data – кодовое слово.

UML-диаграмма данного класса представлена на рисунке 16.

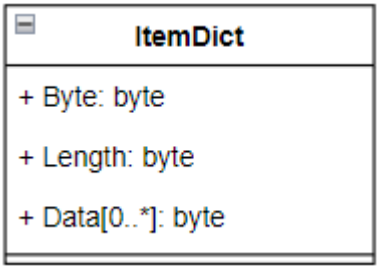

Рисунок 16 – UML-диаграмма класса ItemDict

3.14 Класс PDict

Класс содержит следующие поля:

- public byte Length – количество элементов;
- public List Data – элементы.

Класс также включает следующие методы:

- public void Add(byte Byte, byte Length, byte[] Data) добавляет в словарь элемент;
- public void Sort() сортирует словарь;
- public ItemDict Search(byte Byte) ищет элемент по значению байта.

UML-диаграмма данного класса представлена на рисунке 17.

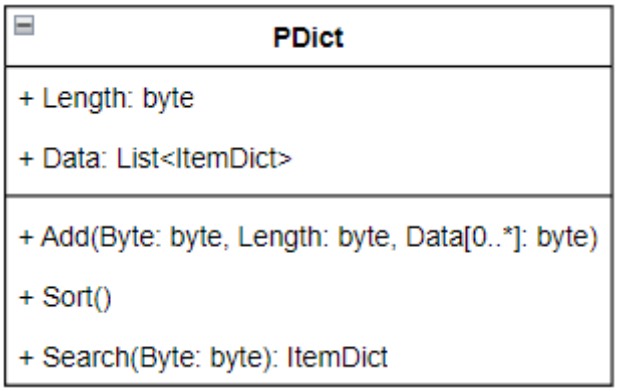

Рисунок 17 – UML-диаграмма класса PDict

3.15 Класс LT

Класс содержит единственное поле private NodeI1O3 _head – начало списка.

Класс также включает следующие методы:

- private bool merge() сдвигает вершину в дерево по алгоритму Хаффмана;
- public void Add(byte value) добавляет байт в список;
- public void Sort() сортирует список по критерию количества встречаемости байтов с одинаковыми значениями;
- public void listToTree() превращает список в дерево;
- public Dict treeToDict() превращает дерево в словарь.

UML-диаграмма данного класса представлена на рисунке 18.

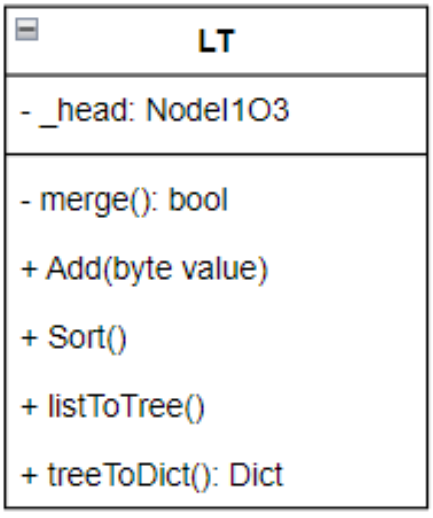

Рисунок 18 – UML-диаграмма класса LT

3.16 Класс Packer

Класс содержит следующие поля:

- private FileStream _inputStream – входной поток;
- private BinaryReader _reader – бинарный поток для чтения;
- private FileStream _outputStream – выходной поток;
- private BinaryWriter _writer – бинарный поток для записи;
- private string _nameArchive – имя архива;
- private string _parent – полное имя каталога, в котором находится архив;
- private LT lt – структура для представления алгоритма Хаффмана;
- private PDict dict – словарь;
- private LinkedList _localNamesArchives – локальные по отношению к корневому архиву имена архивов;
- private LinkedList _localNamesFiles – локальные по отношению к корневому архиву имена файлов;
- private LinkedList _localNamesDirectories – локальные по отношению к корневому архиву имена пустых каталогов;
- private ulong _lengthArchives – количество архивов;
- private ulong _lengthFiles – количество файлов;
- private ulong _lengthDirectories – количество пустых каталогов.

Класс также включает следующие методы:

- public void AddComponent(string localName, FileType type) – добавляет компонент для чтения файла;
- public void Run(){} начинает процесс упаковки;
- private void createComponent(FileStream inputStream) создает компонент для чтения файла;
- private void removeComponent() удаляет компонент для чтения файла;
- private void _readComponent(string path) читает 1 файл(компонент);

- private void Read() читает все файлы;
- private void _generateDict() создает словарь;
- private void _writeLength(ulong length, BinaryWriter writer) записывает длину в выходной поток в динамическом формате (используя префикс как тип длины);
- private void _writeName(string name, BinaryWriter writer) записывает имя файла в выходной поток.

UML-диаграмма данного класса представлена на рисунке 19.

Рисунок 19 – UML-диаграмма класса Packer

3.17 Класс Unpacker_InStream

Класс содержит следующие поля:

- public byte _currentByte – буфер для чтения;
- private FileStream _fileStream – входной файловый поток;
- private BinaryReader _binaryReader – бинарный поток для чтения;
- private ulong length количество битов для распаковки 1 компонента (1 файла после распаковки).

Класс также включает следующие методы:

- public void Create(ulong length) создает компонент чтения;
- public bool End() проверяет достигнут ли конец компонента;
- public byte ReadByte() читает из входного потока 1 байт.

UML-диаграмма данного класса представлена на рисунке 20.

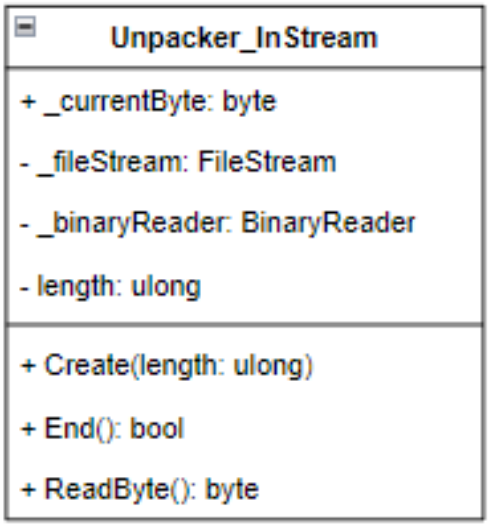

Рисунок 20 – UML-диаграмма класса Unpacker_InStream

3.18 Класс Unpacker_OutStream

Класс содержит единственное поле public BinaryWriter _binaryWriter бинарный поток для записи.

Класс также включает следующие методы:

- public void WriteByte(byte value) записывает 1 байт в выходной поток;
- public void Create(BinaryWriter binaryWriter) создает компонент записи.

UML-диаграмма данного класса представлена на рисунке 21.

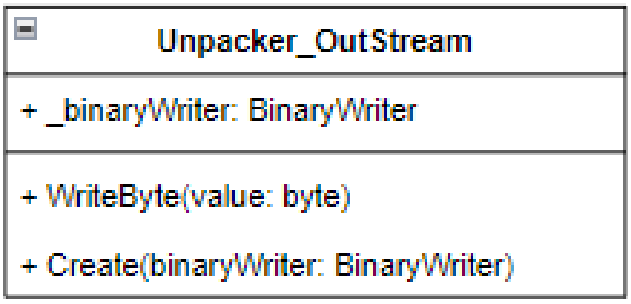

Рисунок 21 – UML-диаграмма класса Unpacker_OutStream

3.19 Класс Unpacker

Класс содержит следующие поля:

- private NodeI1O2 _root – представление словаря бинарным деревом;
- public InStream input – интерфейс для входного потока;
- public OutStream output – интерфейс для выходного потока.

Класс также включает следующие методы:

- public void CreateComponent(string[] partsPath) создает компонент;
- public void RemoveComponent() удаляет компонент;
- public void run() запускает распаковку.

UML-диаграмма данного класса представлена на рисунке 22.

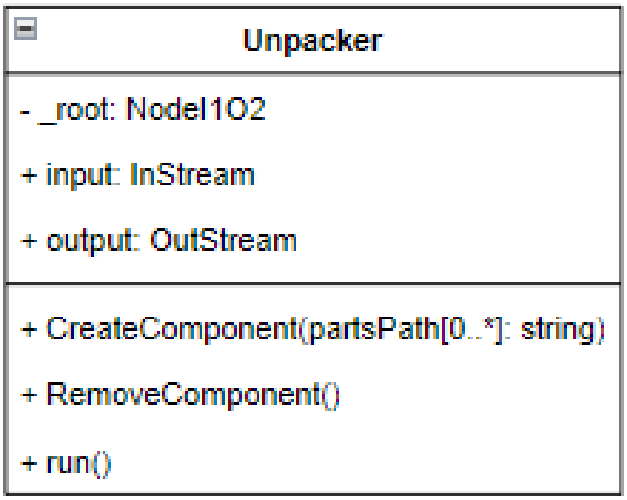

Рисунок 22 – UML-диаграмма класса Unpacker

3.20 Класс MainWindow

Класс содержит следующие поля:

- private int _keySorting – ключ сортировки;
- private bool inArchive – флаг нахождения в архиве;
- private TreeDirectory treeDirectory – каталог внутри архива, представленный деревом;
- private NodeI1OM currentLayerTreeDirectory – родительский узел текущего уровня архива, представленного деревом;
- public ObservableCollection FileList – список файлов на текущем уровне файловой системы;
- public DirectoryInfo CurrentDirectory – родительский каталог;
- public string DirectoryNameArchive – имя архива в самой файловой системе.

Класс также включает следующие методы:

- private void LoadDirectory() – загружает каталог;
- private void _sort() – сортирует загруженный каталог;
- private void fileSystemList_PreviewMouseLeftButtonDown(object sender, MouseButtonEventArgs e) – обработка нажатия ЛКМ;
- private void fileSystemList_MouseDoubleClick(object sender, MouseButtonEventArgs e) – обработка нажатия двойного нажатия;
- private void fileSystemList_PreviewKeyDown(object sender, KeyEventArgs e) – обработка нажатия кнопки на клавиатуре;
- private void OpenUnpackingWindow_Click(object sender, RoutedEventArgs e) – обработка нажатия кнопки распаковки;
- private void OpenPackingWindow_Click(object sender, RoutedEventArgs e) – обработка нажатия кнопки упаковки;
- private void OpenCreateFolder_Click(object sender, RoutedEventArgs e) – обработка нажатия кнопки создания папки.

UML-диаграмма данного класса представлена на рисунке 23.

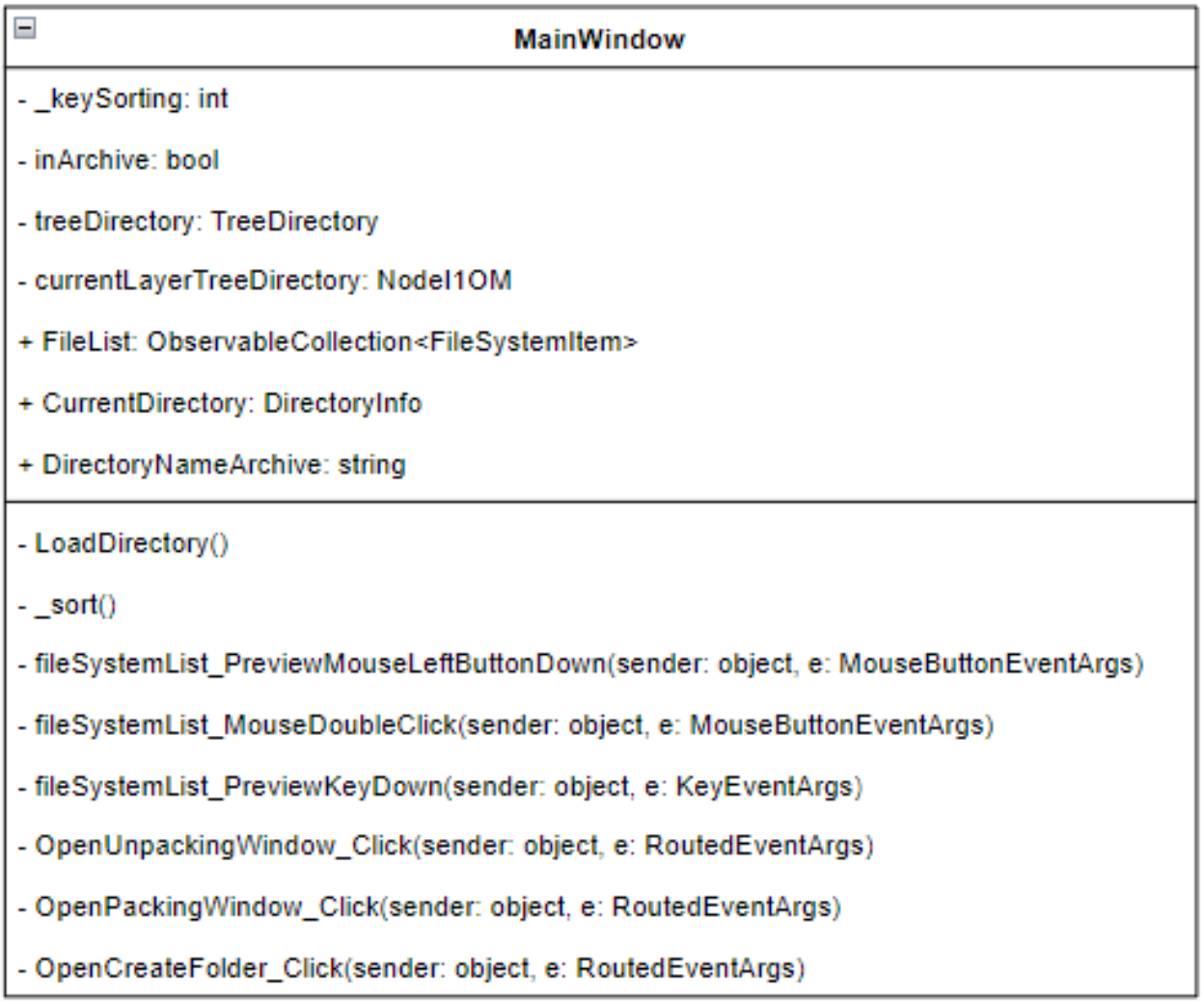

Рисунок 23 – UML-диаграмма класса MainWindow

3.21 Класс PackingWindow

Класс содержит единственное поле private bool _isClickOk – флаг нажатия кнопки начала упаковки.

Класс также включает следующие методы:

- public void Get(out string path, out string name) получает из графического интерфейса путь упаковки и имя архива;
- private void PackingWindow_LocationChanged(object sender, EventArgs e) обрабатывает перемещение окна;
- private void Cancel_Click(object sender, RoutedEventArgs e) обрабатывает нажатие кнопки отмены;
- private void Ok_Click(object sender, RoutedEventArgs e) обрабатывает нажатие кнопки начала упаковки;

- private void Choose_Click(object sender, RoutedEventArgs e) обрабатывает нажатие на кнопку выбора каталога упаковки;
- private void LoadDirectoryTree(string rootDirectory) загружает в графический выбор пути упаковки корневой каталог;
- private void LoadSubdirectories(TreeViewItem parentItem) загружает в графический выбор пути дочерние элементы выбранного элемента;
- private void SubItem_Expanded(object sender, RoutedEventArgs e) обрабатывает нажатие на развернуть/свернуть в графическом выборе пути;
- private void DirectoryTreeView_SelectedItemChanged(object sender, RoutedPropertyChangedEventArgs e) обрабатывает выбор (двойное нажатие) каталога упаковки в графическом выборе пути.

UML-диаграмма данного класса представлена на рисунке 24.

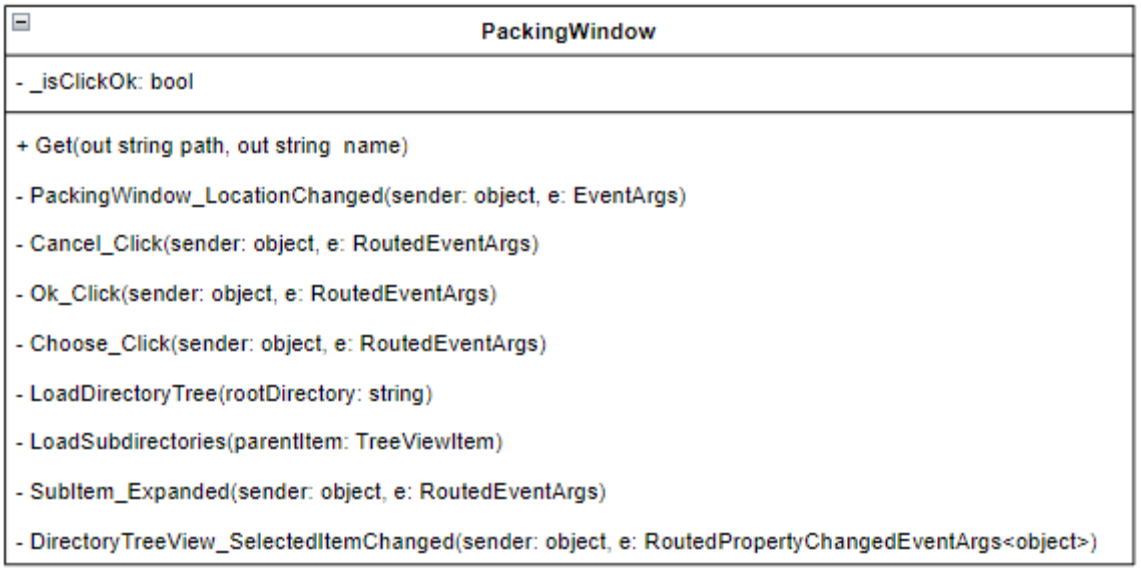

Рисунок 24 – UML-диаграмма класса MainWindow

3.22 Класс UnpackingWindow

Класс содержит следующие поля:

- private List Items – выбранные архивы;
- private TreeDirectory Parent – родительский каталог элементов внутри архива;
- private bool _isClickOk – флаг нажатия начала распаковки.

Класс также включает следующие методы:

- private void UnpackingWindow_LocationChanged(object sender, EventArgs e) обрабатывает перемещение окна;
- private void Cancel_Click(object sender, RoutedEventArgs e) обрабатывает нажатие кнопки отмены;
- private void Ok_Click(object sender, RoutedEventArgs e) обрабатывает нажатие кнопки начала распаковки;
- private void Choose_Click(object sender, RoutedEventArgs e) обрабатывает нажатие на кнопку выбора каталога распаковки;
- private void LoadDirectoryTree(string rootDirectory) загружает в графический выбор пути распаковки корневой каталог;
- private void LoadSubdirectories(TreeViewItem parentItem) загружает в графический выбор пути дочерние элементы выбранного элемента;
- private void SubItem_Expanded(object sender, RoutedEventArgs e) обрабатывает нажатие на развернуть/свернуть в графическом выборе пути;
- private void DirectoryTreeView_SelectedItemChanged(object sender, RoutedPropertyChangedEventArgs e) обрабатывает выбор (двойное нажатие).

UML-диаграмма данного класса представлена на рисунке 25.

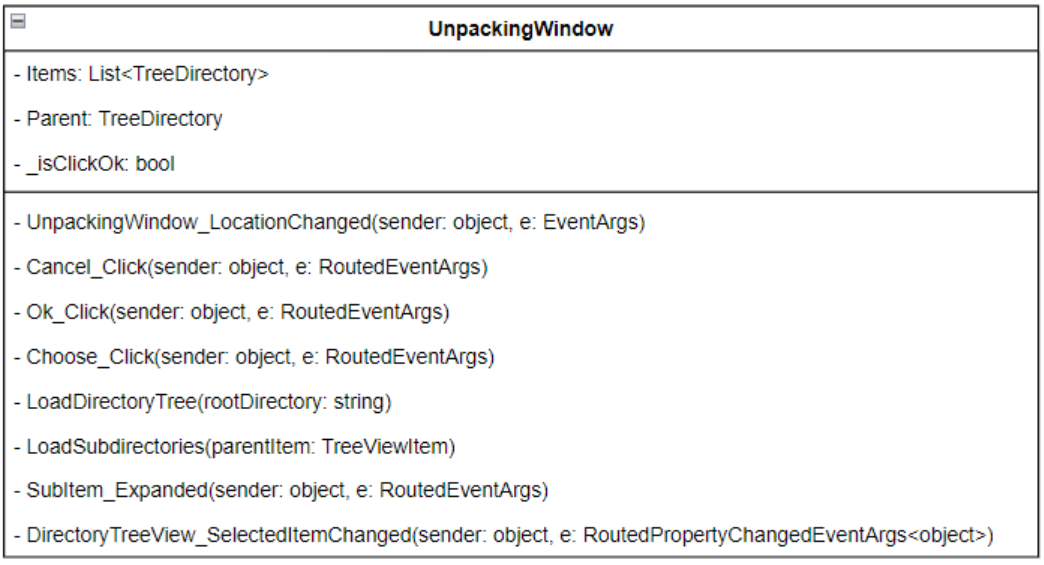

Рисунок 25 – UML-диаграмма класса UnpackingWindow

3.23 Класс CreateFolder

Класс содержит единственное поле private bool _isClickOk – флаг нажатия на кнопку создания папки.

Класс также включает следующие методы:

- public void Get(out string name) получает из графического интерфейса имя папки;
- private void PackingWindow_LocationChanged(object sender, EventArgs e) обрабатывает перемещение окна;
- private void Cancel_Click(object sender, RoutedEventArgs e) обрабатывает нажатие кнопки отмены;
- private void Ok_Click(object sender, RoutedEventArgs e) обрабатывает нажатие кнопки создания папки.

UML-диаграмма данного класса представлена на рисунке 26.

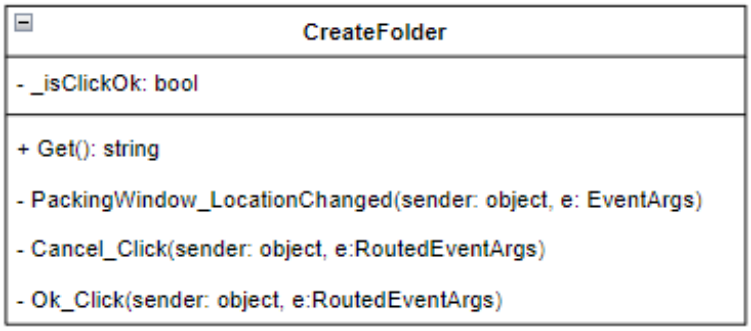

Рисунок 26 – UML-диаграмма класса CreateFolder

3.24 Класс WarningWindow

Класс содержит следующие методы:

- private void WarningWindow_LocationChanged(object sender, EventArgs e) обрабатывает перемещение окна;
- private void ClickButton_OK(object sender, RoutedEventArgs e) обрабатывает нажатие кнопки «OK».

UML-диаграмма данного класса представлена на рисунке 27.

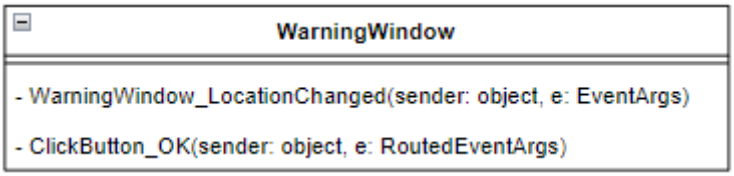

Рисунок 27 – UML-диаграмма класса WarningWindow

3.25 Демонстрация иерархии классов

Разработанная иерархия с абстрактным классом BinaryElement

представлена на рисунке 28.

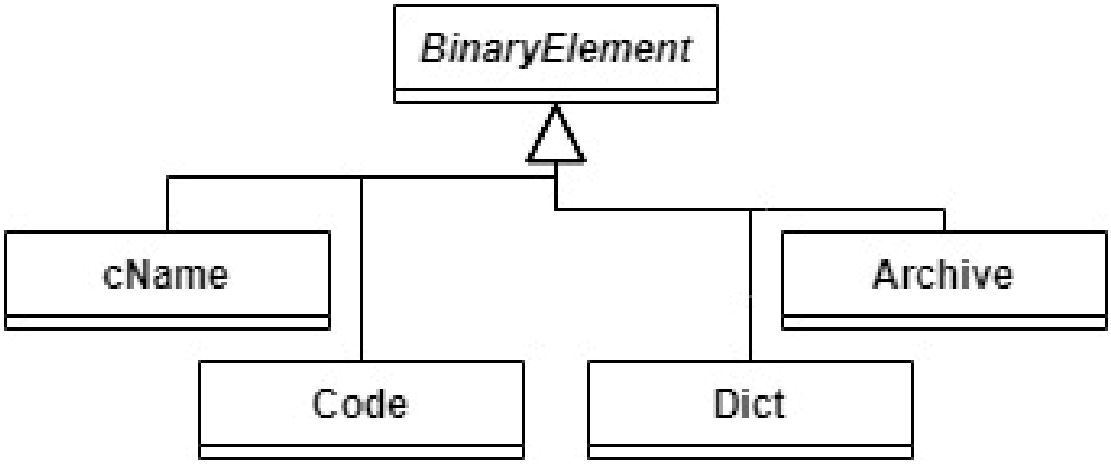

Рисунок 28 – UML-диаграмма иерархии «Бинарный элемент»

Для элемента файловой системы также предусмотрена иерархия в соответствии с рисунком 29.

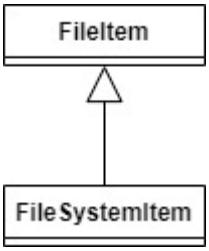

Рисунок 29 – UML-диаграмма иерархии «Элемент файловой системы»

Все окна принадлежат иерархии окна, в соответствии с рисунком 30.

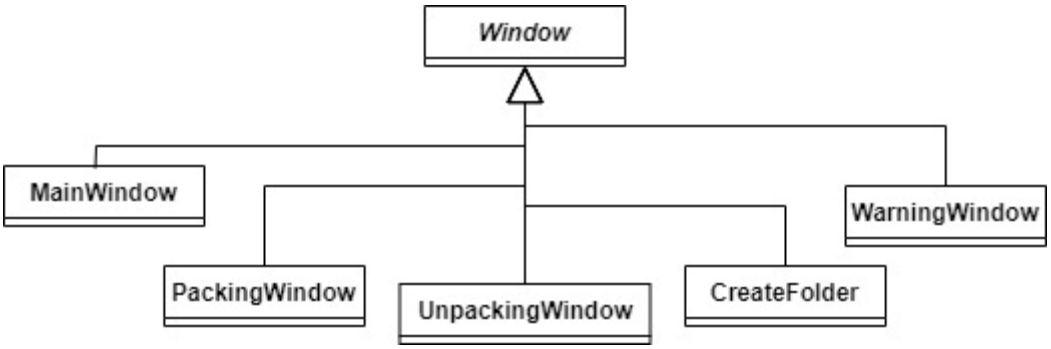

Рисунок 30 – UML-диаграмма иерархии «Окно»

Разработанные иерархии классов позволяют реализовать поставленную задачу в соответствии со сформулированными требованиями.

4 Использованные мультимедийные ресурсы и сторонние библиотеки

В процессе выполнения работы были использованы следующие ресурсы:

- zip.png – иконка приложения, отображаемая при его работе [2];
- add-folder.png – изображение для кнопки «Создать папку» [3];
- inbox.png – изображение для кнопки «Поместить» [4];
- outbox.png – изображение для кнопки «Извлечь» [5];
- box.png – иконка для отображения в файловой системе архивов *.firsov-archive [6];
- file.png – иконка для отображения в файловой системе файлов [7];
- folder.png – иконка для отображения в файловой системе папок [8];
- upload.png – иконка для отображения в файловой системе родительской папки [9].

Использование ресурсов позволило реализовать пользовательский интерфейс и значительно упростить процесс разработки программы.

5 Демонстрация работы

После запуска программы открывается каталог по прописанной заранее директории, в данном случае, как показано на рисунке 31, это вложенная папка на рабочем столе. Сверху располагается панель из 3 возможных действий, ниже файловый менеджер с информацией о файлах внутри текущего каталога.

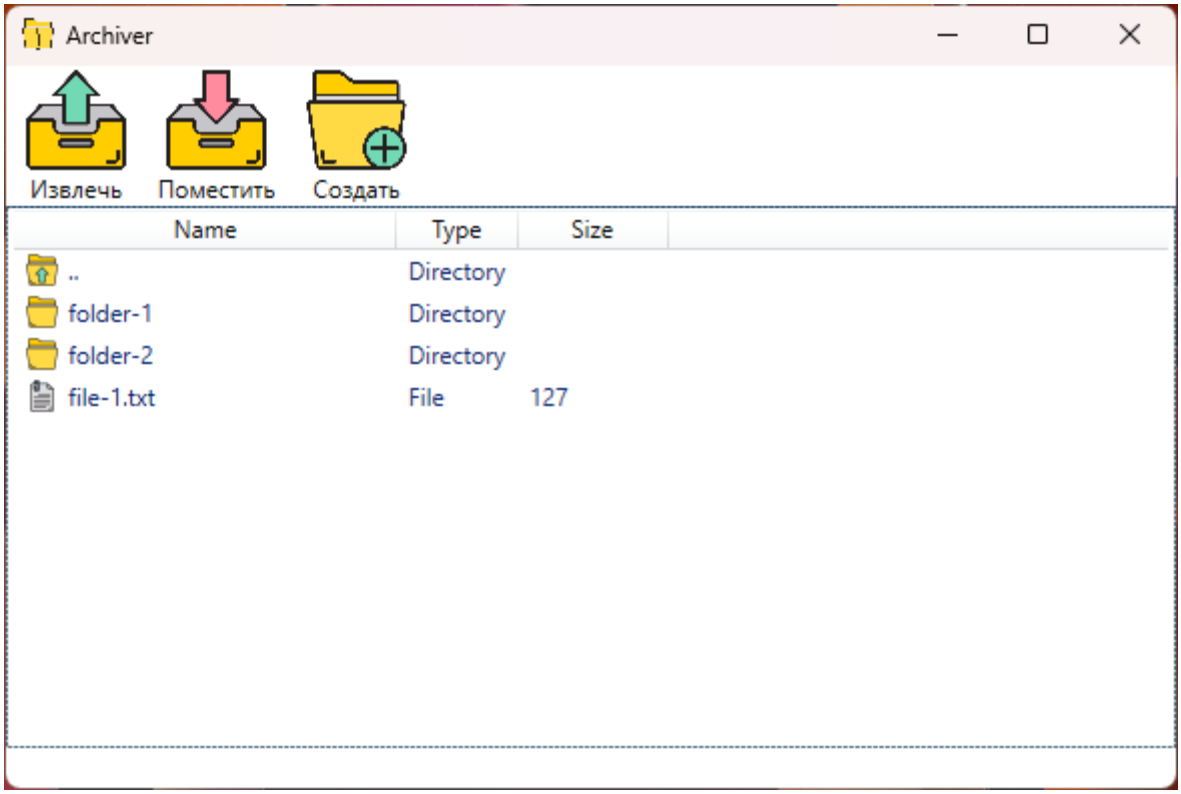

Рисунок 31 – Основное окно программы после открытия

Файл file-1.txt является текстовым, он содержит текст, отображенный на рисунке 32.

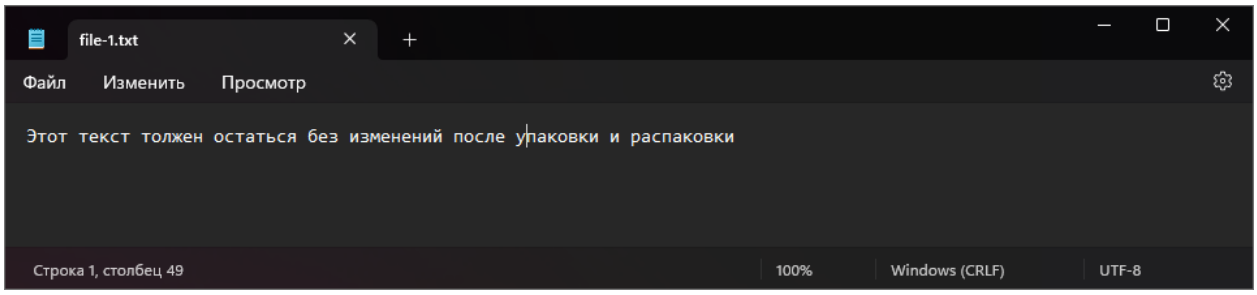

Рисунок 32 – Скриншот содержимого файла file-1.txt

Двойное нажатие по папке или выбор через кнопку Return, позволяет переместиться в нее, результат выбора папки folder-2 отображен на рисунке 33.

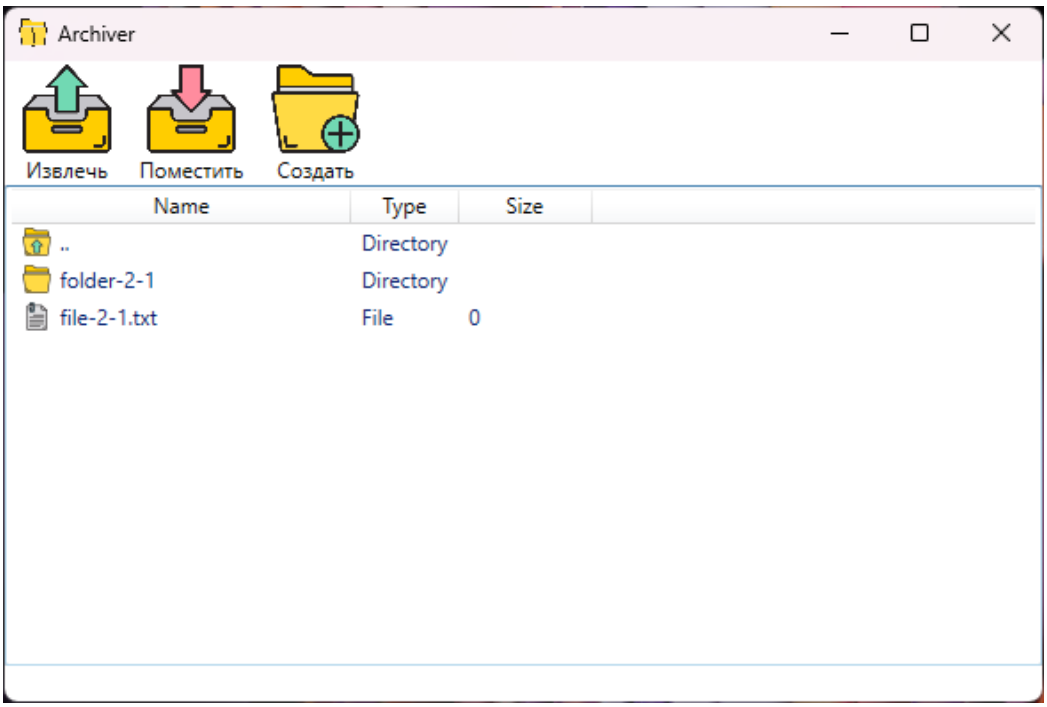

Рисунок 33 – Результат открытие папки

Нажатие на название столбца позволяют устанавливать на него ключ сортировки. Нажатия подряд меняют порядок сортировки на противоположный. Так для варианта, показанного на рисунке 31, после нажатия на заголовок Name список файлов будет отсортирован по имени. На рисунке 34 отображен получившийся список, и анимация нажатия на заголовок.

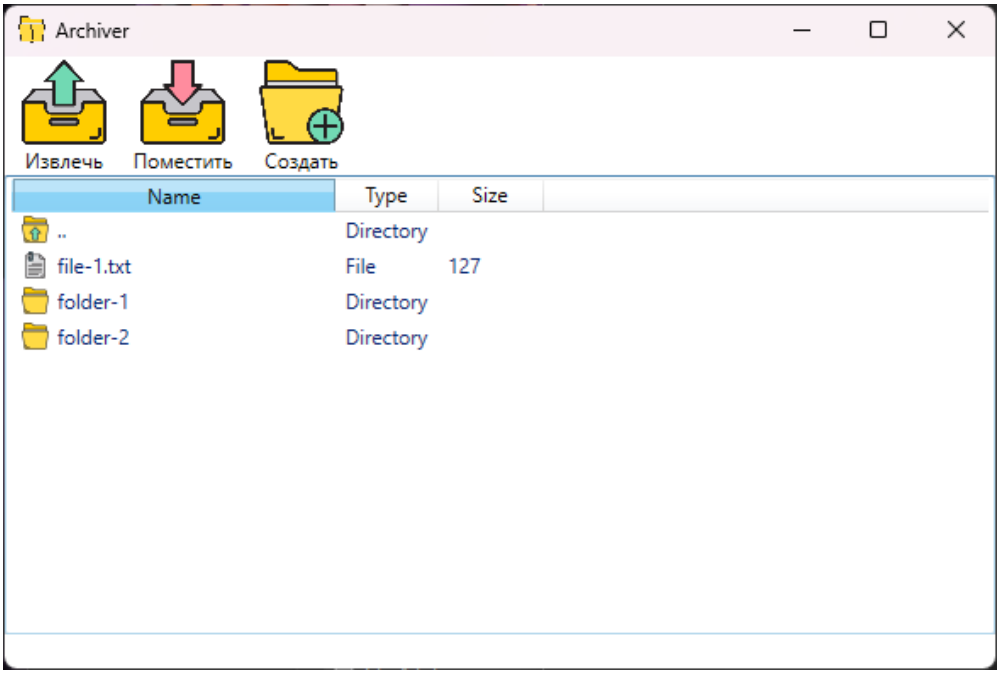

Рисунок 34 – Результат нажатия на заголовок Name

После выбора папки folder и нажатия на «Поместить» отображается окно, открытое в модальном режиме, показанное на рисунке 35, в котором можно выбрать имя архива и путь его сохранения, по умолчанию имя архива – имя выбранного файла, а директория сохранения является путь каталога, в котором расположен сам файл.

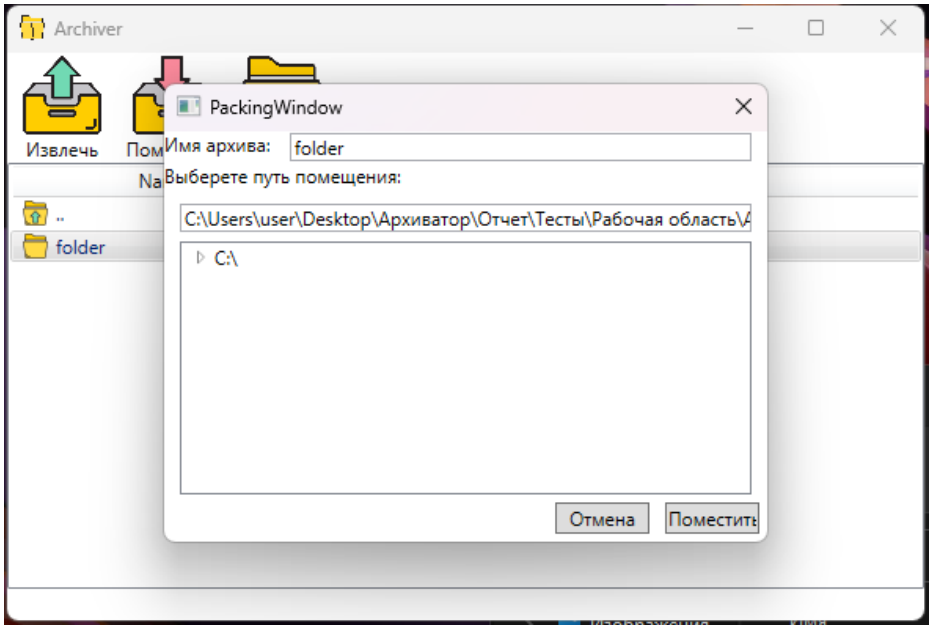

Рисунок 35 – Окно упаковщика

Выбрать путь можно раскрывая списки, начиная с корневого после двойного нажатия, директория будет отображаться в соответствующем окне, как показано на рисунке 36.

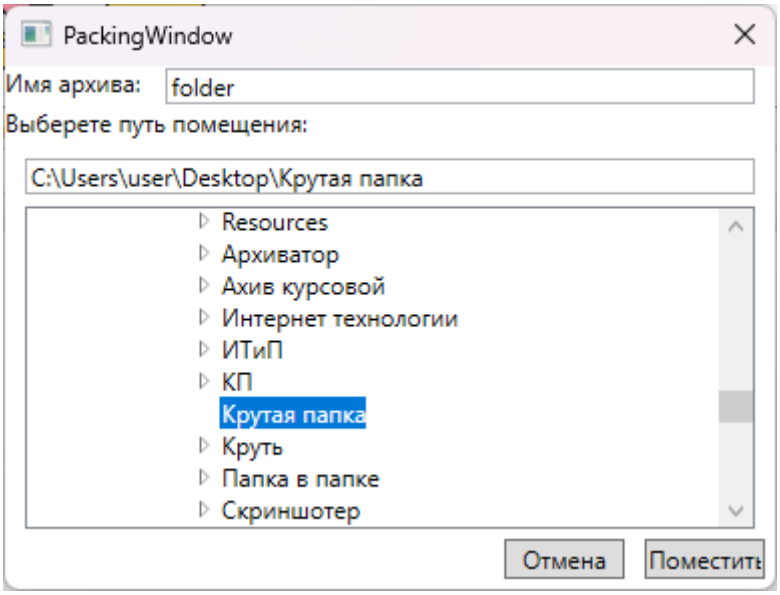

Рисунок 36 – Выбор каталога за счет поочередного раскрытия списка файлов

При нажатии на крестик или «Отмена» окно закроется, если в окне из рисунка 35 нажать «Поместить», создастся архив рядом с выбранной папкой, как показано на рисунке 37.

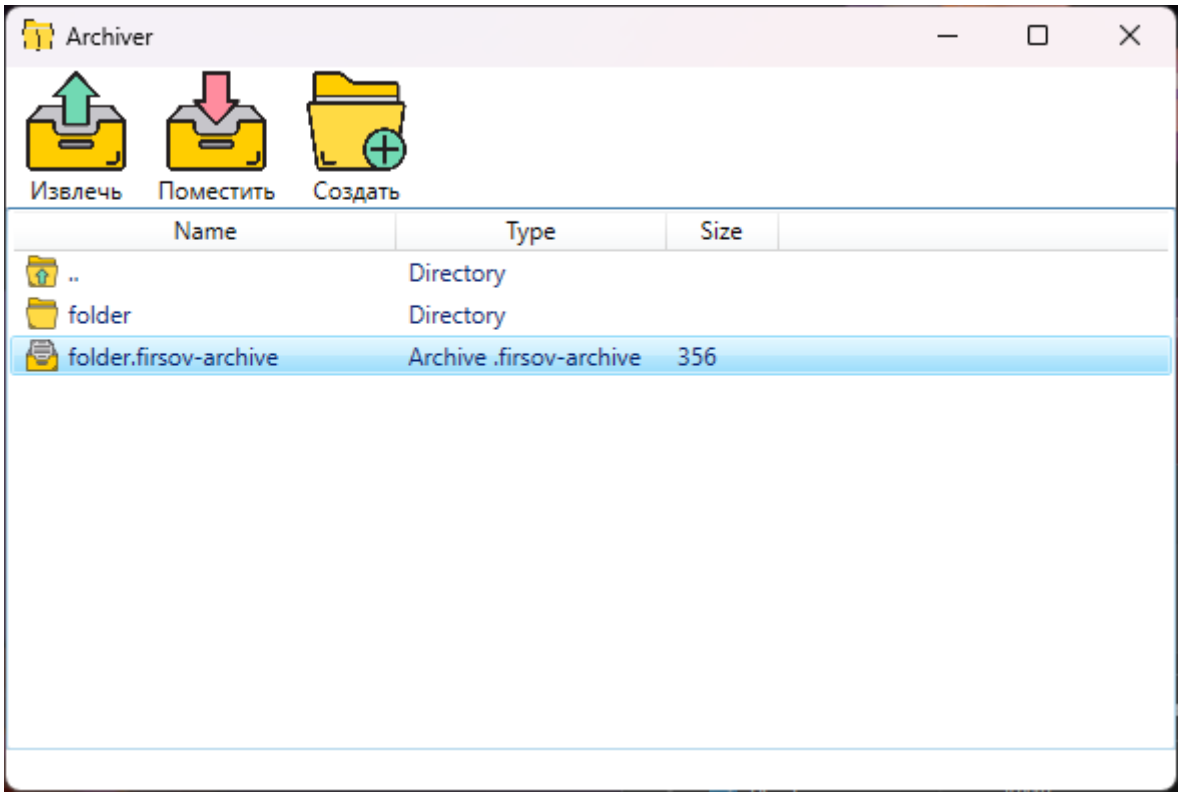

Рисунок 37 – Результат упаковки

В данном случае размер почти в 3 раза больше информации зашифрованной информации. Это происходит, потому что при маленьком размере исходных данных размер служебных данных превышает размер основных данных. Такая же ситуация может возникнуть и при большом размере исходных данных, если они состоят из множества маленьких по размеру файлов (сжатие данных не выигрывает места для записи имени для каждого файла).

После нажатия на архив, его можно открыть без распаковки для предпросмотра, при этом можно открывать папки, но нельзя открывать файлы. После открытия архива, выбора в нем текстового файла и нажатие на «Извлечь», открывается модальное окно, как показано на рисунке 38. Его функционал с точки зрения графического интерфейса ничем не отличается от модального окна, продемонстрированного на рисунке 35, за исключением

отсутствия выбора имени, так как с точки зрения распаковки, директория сохранение и корневая папка для извлеченных файлов одно и то же.

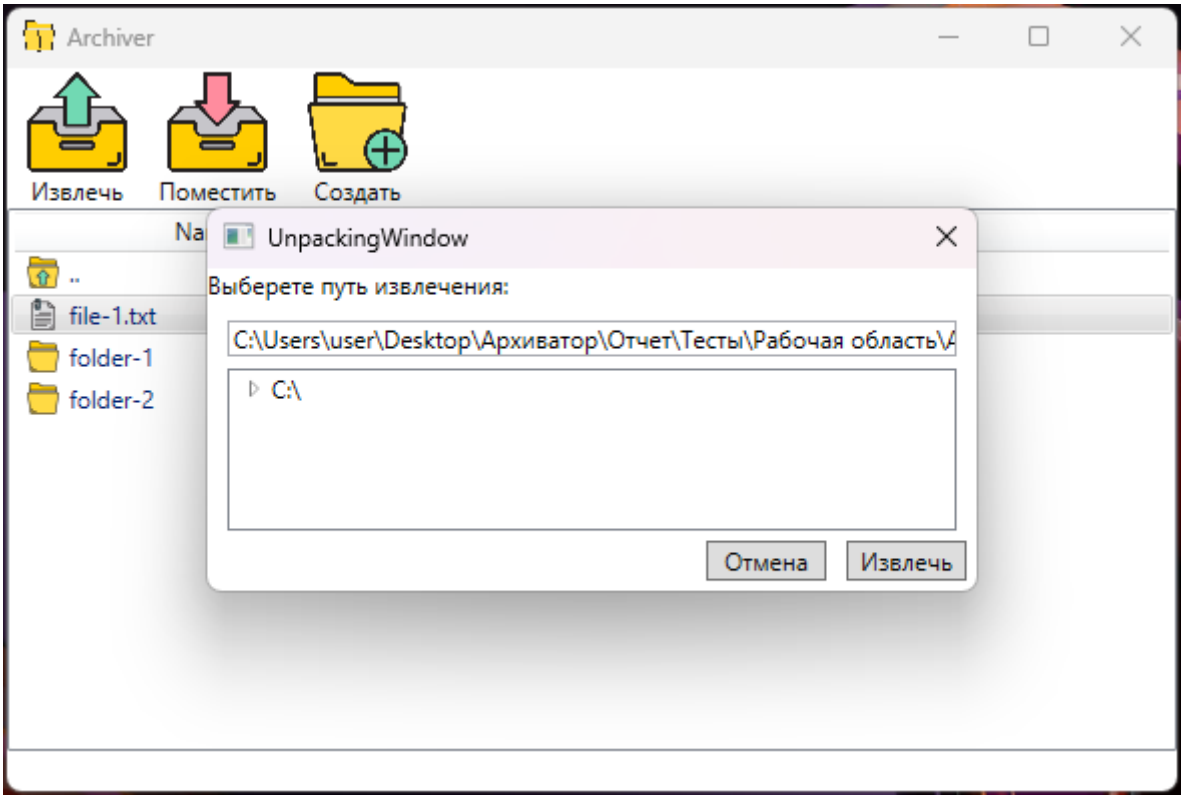

Рисунок 38 – Окно распаковщика

На рисунке 39 отображено содержимое распакованного файла.

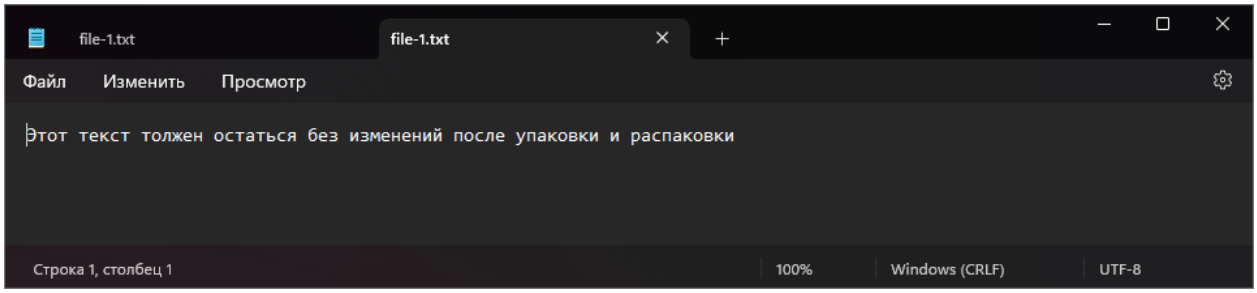

Рисунок 39 – Скриншот содержимого файла file-1.txt после упаковки и распаковки

Этот текст совпадает с исходным до упаковки и распаковки, следовательно архивация происходит без потери данных.

Лучшего сжатия можно достигнуть, если словарь содержит только 1 элемент. Для этого все байты файла должны быть одинаковые. На рисунке 40 показано, в таком случае достигается сжатие, почти в 8 раз, то есть примерно 13%.

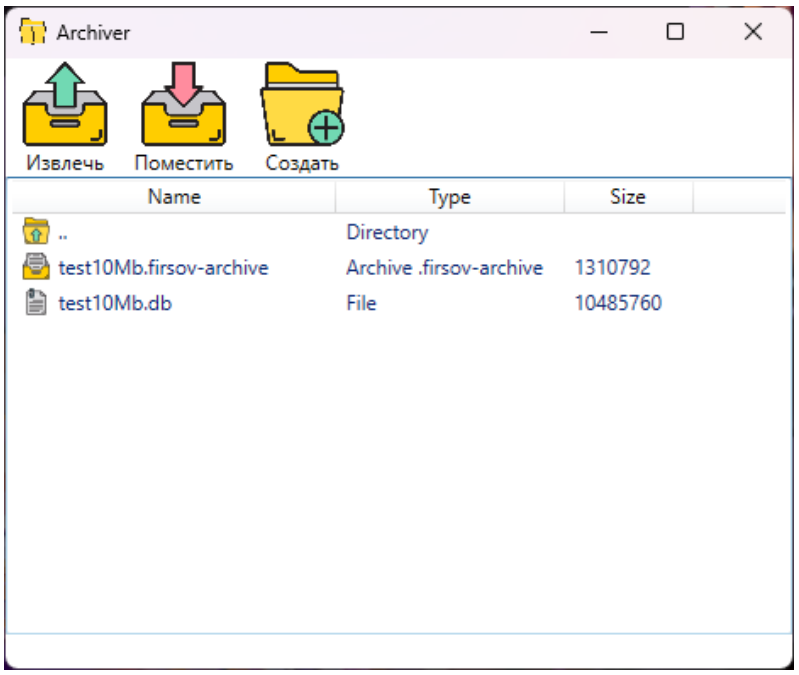

Рисунок 40 – Результат упаковки бинарного файла 10 Мб

На рисунке 41 показан результат упаковки 2 картинок, с большой палитрой и маленькой.

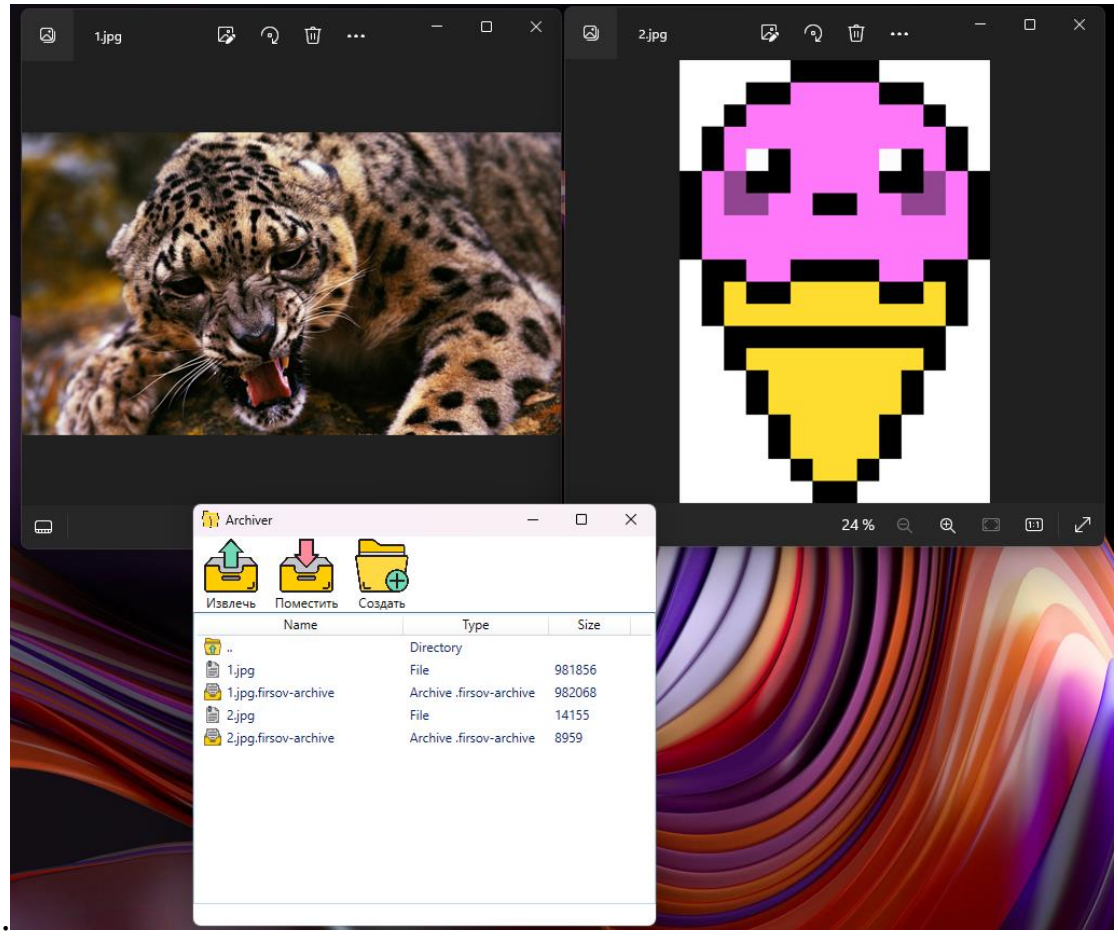

Рисунок 41 – Результат упаковки растровых изображений

Для картинки с большой палитрой сжатие не было достигнуто, в то время как картинки с маленькой оно составило почти 63%.

При кодировании реального текста (не случайного набора символов) за счет того, что используется не все символы кодировки, сжатие происходит эффективно. Так на рисунке 42 продемонстрировано сжатие почти 50%.

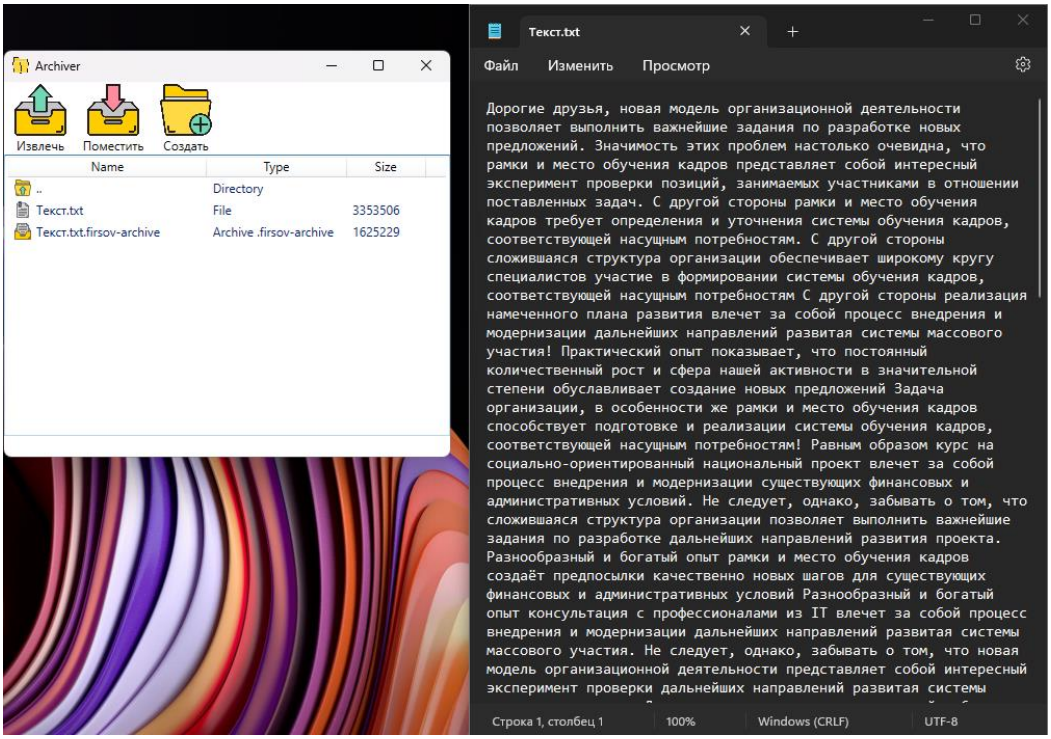

Рисунок 42 – Результат упаковки реального текста

Если выбрать не архив или файл не внутри архива и нажать на «Извлечь», появится окно ошибки, открытое в модальном режиме, как показано на рисунке 43, с соответствующим текстом.

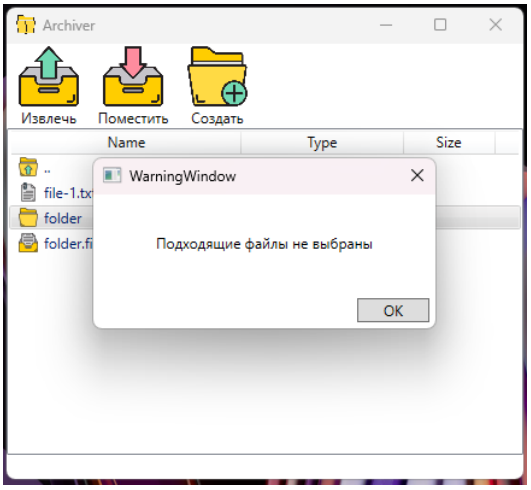

Рисунок 43 – Окно ошибки при неверных действиях при распаковке

Такое же окно будет отображаться если внутри архива попытаться упаковать файлы, как показано на рисунке 44.

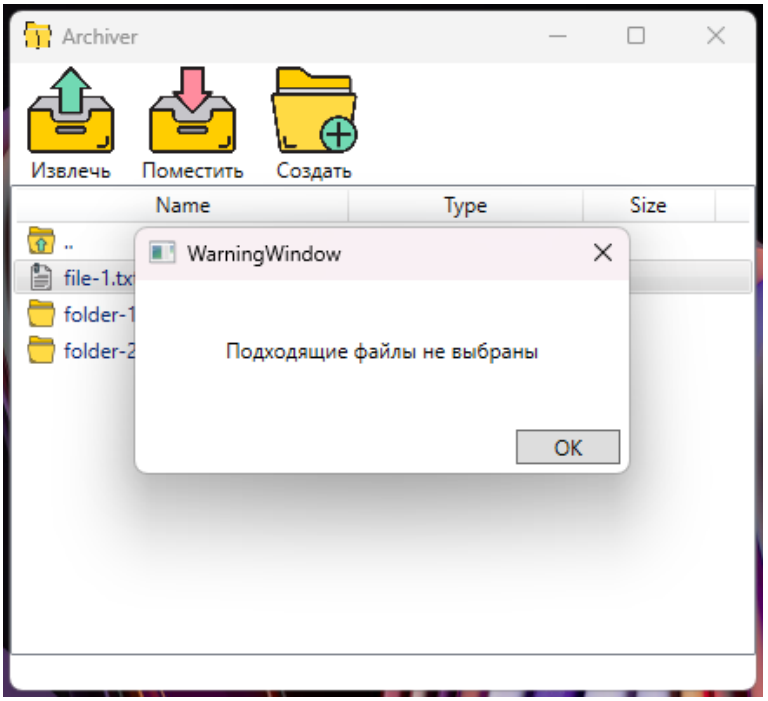

Рисунок 44 – Окно ошибки при неверных действиях при упаковке

Приложение так же включает в себя примитивную функцию файлового менеджера, а именно создания папки. После нажатие на кнопку создать, открывается окно создание папки, открытое в модельном режиме, как показано на рисунке 45, в котором можно ввести имя папки и создать ее.

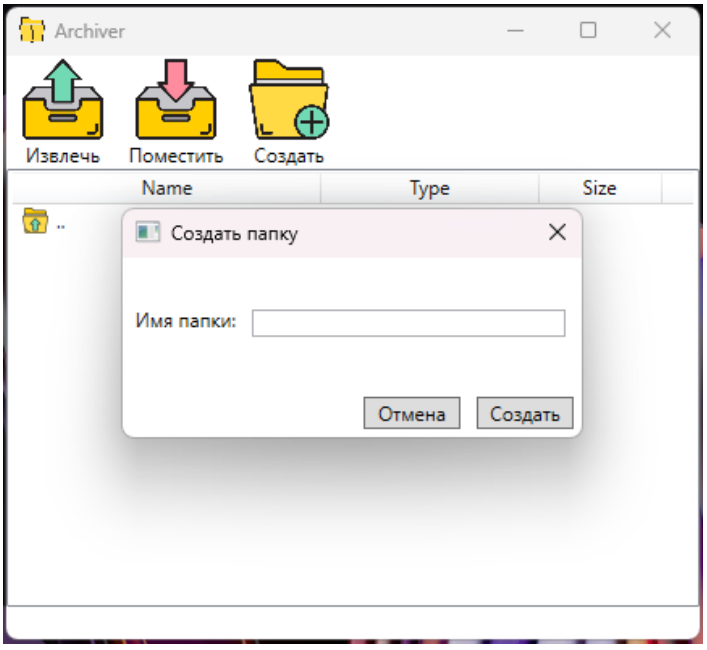

Рисунок 45 – Окно создания папки

На рисунке 46 показана инструкция создания папки с именем «Папка», выполненная 2 раза, причем во 2-й раз кнопка создания, не была опущена.

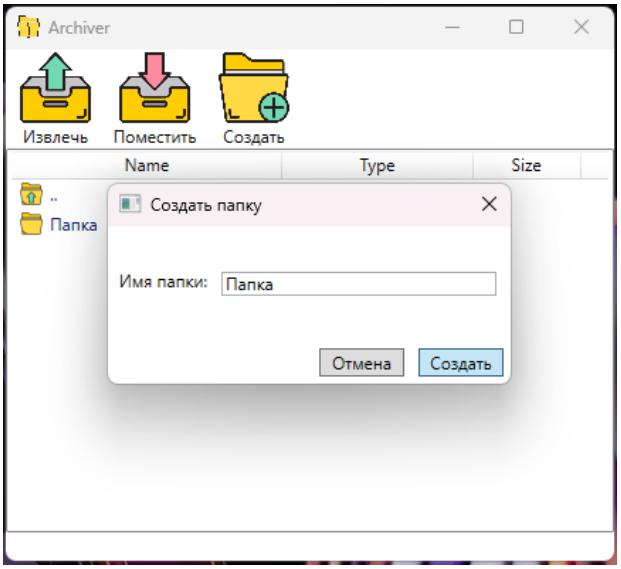

Рисунок 46 – Заполненное окно создания папки

Так как папка с таким именем уже есть, то новая не создается, а вместо нее отображается окно ошибки с соответствующим текстом, как показано на рисунке 47.

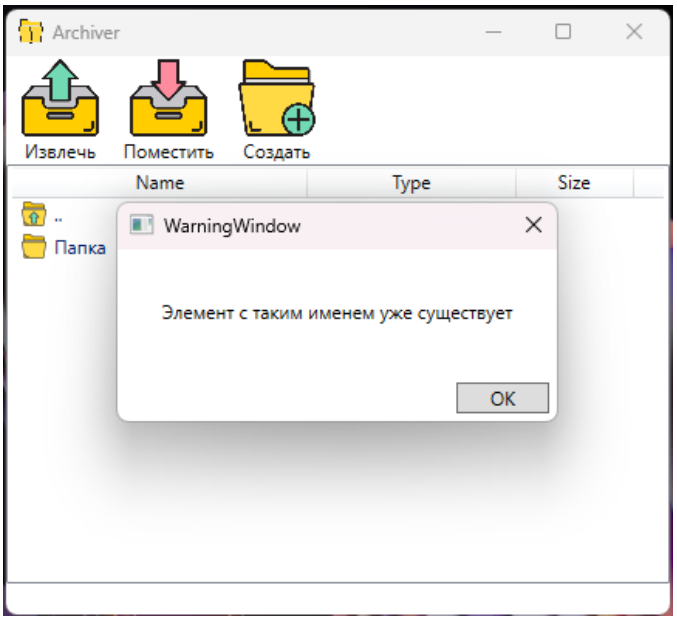

Рисунок 47 – Окно ошибки при неверных действиях при создании папки

Демонстрация возможностей приложения подтверждает его соответствие выдвинутым требованиям и поставленной задачи.

ЗАКЛЮЧЕНИЕ

В результате выполнения курсовой работы был создан архиватор и графический интерфейс к нему. Программа написана на C# с использованием WPF (исходные тексты программы приведены в приложении А).

В ходе выполнения были выполнены следующие задачи:

- описаны основные требования к разрабатываемому приложению;
- составлена иерархия классов, реализованы все требуемые поля и методы;
- разработано приложение, реализованы все указанные окна и функции;
- продемонстрирована работоспособность приложения.

После описания модели архиватора он был написан без использования вспомогательных библиотек. Упаковщик сжимает файлы, а распаковщик извлекает, процесс архивации проходит без потери данных.

Графический интерфейс выполняет примитивные функции файлового менеджера: создание пустых папок и перемещение по каталогу.

Дополнительно добавлены кнопки, запускающие упаковку и распаковку. Взаимодействие пользователя с интерфейсом осуществляется при помощи клавиатуры и мыши.

Алгоритм Хаффмана подразумевает разбиение последовательности на части с последующей перекодировкой каждой на основе количества встречаемости данной последовательности. При таком подходе входные данные будут читаться 2 раза. В данной курсовой работе последовательность разбивалась на байты. В этом случае эффективность сжатия будет достигаться только в случае неполного заполнения словаря или если существует часть, которая используется намного реже остальной. В случае реальных данных эффективно сжиматься по алгоритму Хаффмана будут текстовые файлы, а неэффективно файлы растровых изображений, музыки и т.д.

Все задачи курсовой работы решены, цель – достигнута.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Алгоритм Хаффмана / StudMe. – URL: https://studme.org/187066/informatika/algoritm_haffmana (дата обращения 31.12.2023).

2. Zip / FreePik. – URL: https://ru.freepik.com/icon/zip_1265910#fromView=resource_detail&position=0 (дата обращения 31.12.2023).

3. Add Folder / FreePik. – URL: https://ru.freepik.com/icon/add-folder_1265876#fromView=resource_detail&position=33 (дата обращения 31.12.2023).

4. Inbox / FreePik. – URL: https://ru.freepik.com/icon/inbox_1265899#fromView=resource_detail&position=11 (дата обращения 31.12.2023).

5. Outbox / FreePik. – URL: https://ru.freepik.com/icon/outbox_1265903#fromView=resource_detail&position=7 (дата обращения 31.12.2023).

6. Box / FreePik. – URL: https://ru.freepik.com/icon/inbox_1265891#fromView=resource_detail&position=18 (дата обращения 31.12.2023).

7. File / FreePik. – URL: https://ru.freepik.com/icon/attach_1265877#fromView=resource_detail&position=32 (дата обращения 31.12.2023).

8. Folder / FreePik. – URL: https://ru.freepik.com/icon/folder_1265898#fromView=resource_detail&position=12 (дата обращения 31.12.2023).

9. Upload / FreePik. – URL: https://ru.freepik.com/icon/upload_1265909#fromView=resource_detail&position=1 (дата обращения 31.12.2023).

ПРИЛОЖЕНИЕ А

Текст программы

Исходные тексты программы располагаются на прилагаемом электронном носителе.