Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н. Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)

ФАКУЛЬТЕТ ИУ «Информатика и системы управления»

КАФЕДРА ИУ-7 «Программное обеспечение ЭВМ и информационные технологии»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

НА ТЕМУ:

«Метод сжатия статических изображений без потерь на основе алгоритма Хаффмана»

Студент ИУ7-42М _____ К. Э. Ковалец
(Группа) (Подпись, дата) (И. О. Фамилия)

Руководитель ВКР _____ Н. В. Новик
(Подпись, дата) (И. О. Фамилия)

Нормоконтролер _____ Д. Ю. Мальцева
(Подпись, дата) (И. О. Фамилия)

2025 г.

РЕФЕРАТ

Рассчетно-пояснительная записка к выпускной квалификационной работе «Метод сжатия статических изображений без потерь на основе алгоритма Хаффмана» содержит **??** страниц, 4 части, рисунков, 5 таблиц и список используемых источников из 26 наименования.

Ключевые слова: сжатие изображений, алгоритм Хаффмана, дерево Хаффмана алгоритм LZW, сжатие без потерь, статические изображения, bmp-файлы.

Объект разработки — метод сжатия статических изображений без потерь.

Цель работы: разработать метод сжатия статических изображений без потерь на основе алгоритма Хаффмана.

В первой части работы рассмотрены основные методы сжатия данных без потерь. Сформулированы критерии сравнения методов сжатия. Выполнен сравнительный анализ исследуемых методов по выделенным критериям. Описана формальная постановка задачи в виде IDEF0-диаграммы.

Во второй части разработан метод сжатия статических изображений на основе алгоритма Хаффмана. Описаны основные особенности предлагаемого метода. Изложены ключевые этапы метода в виде схем алгоритмов.

В третьей части обоснован выбор программных средств для реализации предложенного метода. Описан формат входных и выходных данных. Разработано программное обеспечение, реализующее описанный метод. Описано взаимодействие пользователя с программным обеспечением.

В четвертой части в рамках исследования проведено сравнение разработанного метода сжатия статических изображений без потерь с рассмотренными аналогами. В качестве критериев сравнения использовались полученная степень сжатия файла и размер информации, необходимой для распаковки изображения.

Разработанный метод сжатия статических изображений без потерь может применяться в системах хранения и передачи данных, где важна высокая степень сжатия изображений без потери их качества.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Методы сжатия статических изображений активно применяются для хранения и передачи растровых изображений. За счет уменьшения размера файла, методы сжатия позволяют достичь увеличения скорости передачи данных, а также уменьшения занимаемого на диске места.

Например, одной из областей применения методов сжатия изображений является медицина, где важно передавать снимки КТ, МРТ, УЗИ, рентгеновские снимки в медицинских информационных системах с привлечением минимальных ресурсов.

Также методы сжатия изображений активно применяются в интернет-магазинах, где скорость загрузки снимков товаров на странице с ассортиментом является ключевой.

Задача сжатия изображений остается актуальной, так как файлов с ними с каждым днем становится все больше. Совершенствование методов сжатия и разработка новых алгоритмов остается важной задачей для обеспечения эффективного хранения и передачи изображений.

Целью выпускной квалификационной работы является разработка метода сжатия статических изображений без потерь на основе алгоритма Хаффмана.

Для достижения поставленной цели необходимо выполнить следующие задачи:

— провести аналитический обзор известных методов сжатия статических изображений;

— разработать метод сжатия статических изображений без потерь на основе алгоритма Хаффмана;

— разработать программное обеспечение для демонстрации работы созданного метода;

— провести сравнение разработанного метода с аналогами по степени сжатия изображений.

1 Аналитическая часть

1.1 Анализ предметной области

Все изображения можно разделить на две группы.

1. **Статические изображения** — это визуальные представления, не содержащие анимации или элементов взаимодействия. Они остаются неизменными и могут быть представлены в виде рисунков, фотографий, диаграмм или любых других неподвижных изображений сцен или объектов.

2. **Динамические изображения** — это визуальные представления, содержащие движения или изменения в течение времени. В отличие от статических, они могут быть анимированными, интерактивными (реагировать на клики или наведение) или включать переходы. Примеры: анимация, видео, GIF-файлы, интерактивная графика.

Существуют два основных типа сжатия изображений — с потерями и без потерь [**book-1**]. Сжатие без потерь позволяет уменьшить размер файла с сохранением всех деталей исходного изображения. Сжатие с потерями приводит к более сильному уменьшению размера исходного файла ценой потери его деталей, следствием чего является потеря качества изображения.

Разработаны различные форматы файлов изображений, каждый из которых предназначен для определенных задач и использует в своей реализации собственные способы сжатия и сохранения информации. Например, формат JPEG используется для сжатия с потерями, тогда как формат PNG предназначен для сжатия изображений без потерь [**book-2**].

Уменьшить размер сжатого файла можно путем изменения параметров сжатия, таких как качество изображения и его разрешение. Более высокие значения этих параметров приведут к увеличению размера получеого файла.

Сокращение числа цветов в изображении также будет способствовать уменьшению размера сжатого файла, так как приведет к снижению числа байт, необходимых для представления каждого пикселя [**article-algorithms**].

1.2 Методы сжатия статических изображений без потерь

1.2.1 Метод RLE

Метод RLE [**RLE**] основан на идее кодирования последовательностей повторяющихся значений. Задача данного метода заключается в нахождении цепочек одинаковых символов и замене их на одно значение из последовательности и количество повторений. Такой алгоритм подходит для сжатия как текстовых файлов, так и изображений, где в роли каждого символа выступает набор байт, необходимый для хранения одного пикселя.

Например, если изображение содержит несколько подряд идущих пикселей одного цвета, то рассматриваемый метод закодирует данную последовательность в виде одного пикселя и количества его повторений. Из описания алгоритма можно сделать вывод о том, что он эффективен для изображений с большим количеством повторяющихся участков или частыми областями одного цвета. Однако RLE не подходит для изображений с большим количеством деталей, где, помимо низкой степени сжатия, может привести к увеличению размера файла.

Рассматриваемый метод относится к сжатию без потерь и может использоваться как отдельно, так и в комбинации с другими алгоритмами для достижения лучшего результата.

Процесс работы алгоритма можно описать следующим образом [**article-rle**].

1. Создать пустую строку для хранения результата.
2. Произвести проход по всем пикселям изображения (по всем строкам слева направо).
3. Для каждого пикселя.
   - Если он совпадает с предыдущим:
     - увеличить счетчик повторений.
   - Если отличается или достигнут конец строки:
     - в результирующую строку записать значение предыдущего пикселя и количество его повторений;

– сбросить счетчик и начать отсчет заново.

4. После завершения прохода добавить данные о последнем пикселе и количестве его повторений.

5. Вернуть строку с закодированными данными.

Пример работы алгоритма:

— входная строка: `FFFFCCBBBDAA`;

— закодированные данные: `4F2C3B1D2A`.

1.2.2 Словарные методы

Словарные методы сжатия, например LZW (Lempel-Ziv-Welch) [**LZW**], основаны на использовании специального словаря, в котором повторяющиеся последовательности заменяются более короткими кодами. Такой подход позволяет существенно сократить объем данных за счет замены часто встречающихся шаблонов уникальными кодами из словаря.

Принцип работы алгоритма LZW заключается в том, что он анализирует входные данные и постепенно строит словарь, где каждой повторяющейся последовательности символов присваивается определенный код. После создания словаря эти коды используются для замены изначальных последовательностей, что и обеспечивает сжатие.

Изначально словарь содержит все возможные односимвольные последовательности (пиксель изображения считается за один символ) [**article-lzw**]. Далее LZW начинает проходить по всем значениям входной строки, формируя текущую последовательность символов. Как только сформируется цепочка значений, отсутствующая в словаре:

— текущую последовательность добавится в словарь с уникальным кодом;

— в выходной поток запишется код уже известной цепочки.

Этот процесс продолжается до тех пор, пока весь поток данных не будет обработан. При декодировании используется тот же словарь значений.

Пошаговый алгоритм метода LZW можно описать следующим образом.

1. Создать словарь со всеми возможными односимвольными значениями (пикселями).

2. Установить начальное значение последовательности W равным первому символу входных данных.

3. Последовательно считывать символа K из входного потока.

   — Проверить, содержится ли комбинация $W + K$ в текущем словаре.

   — Если да:

      – присвоить W значение $W + K$.

   — Если нет:

      – добавить код для текущей последовательности W в результат;

      – добавить новую комбинацию $W + K$ в словарь с уникальным кодом;

      – присвоить W значение K.

4. Добавить код последней оставшейся в буфере последовательности W.

5. Вернуть полученный результат в виде последовательности кодов.

1.2.3 Унарное кодирование

Унарное кодирование [**UnaryEncoding**] представляет собой префиксный код, основанный на идее представления целых положительных чисел n в виде последовательности из $(n-1)$ битов со значением 1, и одного бита со значением 0, следующим за цепочкой единиц. Также возможно обратное представление, где в начале кода идет последовательность из $(n-1)$ нулей, заканчивающаяся единицей.

Данный метод следует использовать только для представления очень малых чисел, в противном случае унарное кодирование может привести к увеличению размера исходного файла. Примером эффективного использования является сжатие изображений с малой цветовой палитрой. Например, при сжатии черно-белой фотографии с малым количеством оттенков серого (с 8 оттенками для примера) самый часто встречающийся пиксель получит значение унарного кода `0` (соответствующее $n = 1$), а самый редко встречающийся будет представлен значением `11111110` (соответствующим $n = 8$).

Унарное кодирование редко применяется в чистом виде для сжатия изображений, однако используется в качестве одного из этапов более сложных гибридных методов сжатия.

Примеры унарного кодирования:

— представление числа 1 (первый вариант) — `0`;

— представление числа 2 (первый вариант) — `10`;

— представление числа 3 (первый вариант) — `110`;

— представление числа 4 (первый вариант) — `1110`.

1.2.4 Метод Хаффмана

Метод Хаффмана [**Huffman**] представляет собой метод сжатия данных без потерь, в основе которого лежит замена часто встречающихся символов более короткими кодами, а редко встречающихся — более длинными. При использовании данного метода для сжатия изображений пиксели будут выступать в роли кодируемых символов.

При реализации метода Хаффмана необходимо вычислить вес каждого пикселя на основе вероятности вхождения символов в сообщение. Полученная таблица частот будет использоваться при построении дерева Хаффмана.

Алгоритм построения дерева Хаффмана [**article-huffman**].

1. Создается список узлов из всех уникальных символов сообщения.
2. Каждому узлу присваивается вес, равный частоте появления символа в сообщении.
3. Из списка выбираются два узла с наименьшими весами.
4. Создается новый узел-родитель, объединяющий выбранные два, с весом, равным сумме весов дочерних узлов.
5. В направлении к потомку с меньшим весом назначается бит 1, с большим весом — бит 0.
6. Два потомка удаляются из списка, а на их место ставится новый родительский узел.

7. Шаги 2–5 повторяются до тех пор, пока в списке не останется один узел — корень дерева.

Построенное дерево используется для получения кодов символов (пикселей). Для этого нужно выбрать лист дерева Хаффмана, соответствующий текущему значению, и построить путь от него до корня дерева, добавляя биты при каждом переходе от одного узла к другому. Полученная последовательность битов будет представлять собой код данного символа (пикселя), записанный в обратном порядке.

Благодаря уникальным префиксам полученных кодов, каждый символ может быть однозначно декодирован, несмотря на то, что коды Хаффмана имеют переменную длину.

Для восстановления исходного изображения требуется доступ к таблице частот пикселей, по которой можно воссоздать дерево Хаффмана и декодировать сжатые данные. Хранение такой таблицы или структуры дерева приводит к увеличению размера сжатого файла. Также к недостатку рассмотренного метода относится необходимость в дополнительном проходе по изображению:

— первый — для построения дерева Хаффмана;

— второй — для кодирования данных.

1.2.5 Арифметическое кодирование

Арифметическое кодирование [**ArithmeticCoding**] — это блочный метод сжатия данных, при котором все входное сообщение преобразуется в единый код, уникальный для данной последовательности символов. При таком подходе нельзя разбить код на отдельные части, соответствующие отдельным символам.

В основе метода лежит представление всего сообщения в виде двоичной дроби в диапазоне от 0 до 1. По мере добавления символов интервал, соответствующий текущему сообщению, сужается, требуя все больше битов для точного его описания.

Для работы алгоритма необходимо заранее определить вероятности появления каждого символа. Значения с большей вероятностью появления уменьшают текущий интервал в меньшей степени, в результате чего к окончательному коду добавляется меньше битов. В то время как редкие символы

требуют большего уточнения интервала и, следовательно, увеличивают длину кода.

Каждому символу сопоставляется определенный интервал на числовой оси, длина которого соответствует вероятности его появления. Эти интервалы располагаются последовательно один за другим и в сумме образуют диапазон от 0 до 1. При кодировании каждый новый символ «вырезает» из текущего интервала подинтервал, соответствующий своей вероятности и положению на оси. В результате конечный интервал однозначно представляет все сообщение.

В отличии от метода Хаффмана, арифметическое кодирование не требует дополнительного прохода по входной строке как при кодировании, так и при восстановлении данных.

1.2.6 Сравнение методов сжатия без потерь

Сравнение предлагается проводить по следующим критериям.

1. Возможность кодирования данных за один проход.
2. Отсутствие необходимости в таблице частот пикселей сжимаемого изображения.
3. Наличие в зашифрованном сообщении информации для дешифровщика (распаковщика).
4. Наличие у каждого сжатого пикселя своего кода.

Результаты сравнения методов сжатия изображений без потерь приведены в таблице **??**.

Таблица 1.1 – Сравнение рассмотренных методов сжатия изображений без потерь

Метод сжатия	**Кр. 1**	**Кр. 2**	**Кр. 3**	**Кр. 4**
Алгоритм RLE	+	+	–	–
Словарные алгоритмы	+	+	+	–
Унарное кодирование	+	–	+	+
Алгоритм Хаффмана	–	–	+	+
Арифметическое кодирование	+	–	+	–

1.3 Методы сжатия статических изображений с потерями

1.3.1 Метод сжатия JPEG

JPEG (Joint Photographic Experts Group) [**JPEG**] — это метод сжатия изображений с потерями, широко используемый для хранения полноцветных фотографий. Свою популярность JPEG приобрел благодаря способности эффективно уменьшать объем изображений без значительного ухудшения визуального качества. Данный метод работает с блоками пикселей размером 8×8, яркость и цвет в которых, как правило, изменяются плавно. После разбиения изображения на блоки, каждая полученная матрица раскладывается в двойной ряд по косинусам, что позволяет выделить наиболее значимые коэффициенты. Такой подход позволяет осуществить плавное изменение цветов в изображении, что и приводит к сжатию.

Алгоритм JPEG можно описать следующим образом.

1. **Разделение изображения на блоки** — исходное изображение делится на небольшие квадратные фрагменты по 8×8 пикселей.

2. **Преобразование цветовой модели** — преобразование исходной цветовой модели (чаще всего RGB) в YCbCr, где Y обозначает яркость, а Cb и Cr — цветовые составляющие.

3. **Дискретное косинусное преобразование (DCT)** — применяется к каждому блоку, преобразуя его в набор частотных коэффициентов, отражающих яркостные и цветовые колебания.

4. **Квантование** — полученные коэффициенты округляются с использованием таблицы квантования, в результате чего незначительные детали отбрасываются, что и приводит к потерям.

5. **Энтропийное кодирование** — к оставшимся данным применяется метод энтропийного кодирования, например, алгоритм Хаффмана, где часто встречающиеся значения получают более короткие коды.

6. **Формирование файла** — все закодированные данные объединяются в структуру JPEG-файла, пригодного для хранения и передачи.

7. **Декодирование** — восстановление изображения происходит в обратной последовательности: распаковка, декодирование, обратное квантование, обратное DCT и преобразование в исходную цветовую модель.

Полученное изображение хоть и теряет часть информации, но сохраняет достаточное качество для большинства практических задач.

1.3.2 Wavelet сжатие

Вейвлет-преобразование [**WaveletCompression**] — это метод обработки сигналов, позволяющий анализировать их частотные характеристики. В отличие от алгоритмов сжатия, таких как JPEG или фрактальные методы, вейвлеты не требуют предварительного разбиения изображения на блоки, так как могут применяться ко всему изображению целиком, что помогает избежать появления артефактов на границах блоков.

Например, рассмотрим одномерное преобразование Хаара. Оно работает с парами элементов сигнала, вычисляя их полусумму и полуразность. Эти два значения можно использовать для восстановления исходных данных, что делает преобразование обратимым. В результате сигнал разобьется на:

— приближенную часть (низкочастотную составляющую), содержащую основную информацию;

— уточняющую часть (высокочастотную составляющую), отражающую мелкие детали.

Двумерное вейвлет-преобразование Хаара строится на применении одномерного преобразования сначала к строкам матрицы изображения, затем к его столбцам. В итоге получится четыре подматрицы:

— одна матрица с пониженным разрешением, представляющая приближенное изображение;

— три матрицы, содержащие уточняющие детали по горизонтали, вертикали и диагонали.

Сжатие достигается за счет обнуления наименее значимых коэффициентов из уточняющих подматриц, что позволяет существенно уменьшить объем данных при сохранении важных деталей изображения.

Рассмотренный метод хорошо подходит для сжатия изображений с плавными переходами, что может быть полезно для работы с медицинскими снимками.

1.3.3 Фрактальный метод

Фрактальное сжатие изображений [**FractalCompression**] — это метод, основанный на использовании фрактальных кодов и принципа самоподобия, заключающегося в том, что отдельные участки изображения могут быть приближенно выражены через другие его части.

Ход работы алгоритма можно описать следующим образом.

1. **Разбиение изображения на фрагменты.** Исходное изображение делится на небольшие блоки пикселей (диапазонные блоки), размер которых может составлять 4×4 или 8×8. Каждый из таких блоков впоследствии сравнивается с другими частями изображения для выявления самоподобия.

2. **Определение базисных блоков.** Среди множества участков изображения выбираются опорные блоки (фракталы [**Fractal**]), которые будут использоваться в качестве шаблонов для аппроксимации других квадратов.

3. **Поиск наилучших совпадений.** Для каждого участка изображения ищется наиболее похожий базисный блок. Сходство между блоками оценивается с помощью метрик, таких как среднеквадратичное отклонение или евклидово расстояние.

4. **Применение преобразований.** Для определения точного соответствия между рассматриваемым фракталом и диапозонным блоком, к первому применяются различные преобразования, такие как поворот, масштабирование и отражение.

5. **Формирование фрактального описания.** Для каждого блока изображения сохраняется информация о подобранном фрактале и параметрах его преобразования. Эти данные составляют сжатое представление изображения — фрактальный код.

6. **Декодирование изображения.** Восстановление блоков изображения происходит путем поочередного применения к базисным фракталам сохраненных преобразований.

Фрактальное сжатие следует использовать для изображений, содержащих множество повторяющихся структур, таких как текстуры, элементы природы и архитектуры. Данный метод обладает высокой вычислительной сложностью, что затрудняет его использование для работы с изображениями большого размера.

Пример изображения с фракталами приведен на рисунке **??**.

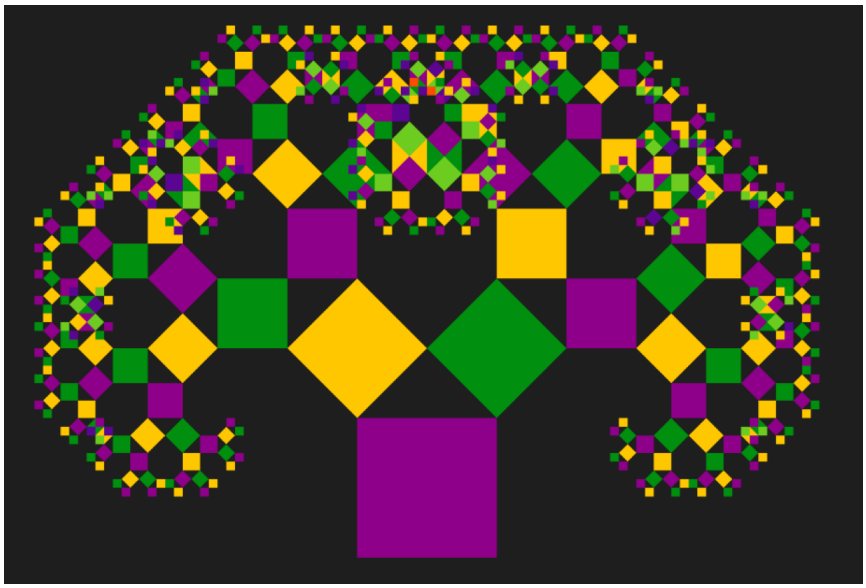

Рисунок 1.1 – Пример изображения с фракталами

1.3.4 Сравнение методов сжатия с потерями

Сравнение предлагается проводить по следующим критериям.

1. Идея, на которой строится алгоритм сжатия.
2. Тип артефактов, возникающих при больших коэффициентах сжатия.
3. Необходимость разбиения исходного изображения на блоки.
4. Необходимость преобразование изображения из цветовой модели RGB в цветовую модель YCbC.

Результаты сравнения методов сжатия изображений без потерь приведены в таблице **??**.

Таблица 1.2 – Сравнение рассмотренных методов сжатия изображений без потерь

Метод сжатия	**Кр. 1**	**Кр. 2**	**Кр. 3**	**Кр. 4**
JPEG	Дискретное косинусное преобразование	Блочные артефакты	+	+
Wavelet	Вейвлет-преобразование	Кольцевые артефакты	–	–
Фрактальный	Самоподобие множеств	Артефакты реконструкции	+	–

1.4 Цветовые модели изображений

1.4.1 Анализ цветовых моделей изображений

Цветовые модели изображений представляют собой математические способы представления цветов в таком формате, в котором цифровые устройства могут их интерпретировать и отобразить, выполнить над ними различные манипуляции или перевести из одного формата в другой. Данные модели определяют структуру хранения цветов. Далее представлены основные цветовые модели изображений.

1. **RGB (Red, Green, Blue) [RGB]** — это аддитивная модель цвета, основанная на смешивании трех основных компонентов: красного, зеленого и синего. Она широко применяется в электронных устройствах, где изображение формируется путем добавления света. Каждый цветовой канал принимает значения от 0 до 255, что дает более 16 миллионов возможных оттенков. RGB считается стандартной моделью для отображения информации на экранах.

2. **RGBA (Red, Green, Blue, Alpha) [RGB]** — это расширенная версия модели RGB, включающая альфа-канал, который задает степень прозрачности. Значение альфа-канала варьируется от 0 (полностью прозрачно) до 255 (полностью непрозрачно). Эта модель особенно полезна в

компьютерной графике и анимации, где требуется реализовать эффекты прозрачности и наложения.

3. **CMYK (Cyan, Magenta, Yellow, Key/Black) [CMYK]** — субтрактивная модель, используемая преимущественно в печатных технологиях. Цвета формируются путём вычитания света при наложении цветных чернил на белую поверхность. В отличие от RGB, CMYK использует голубой, пурпурный, жёлтый и чёрный цвета. Такая модель обеспечивает высокую точность цветопередачи при печати и применяется в типографике и издательском деле.

4. **LAB (Lightness, A, B) [Lab]** — модель, описывающая цвет на основе восприятия человеческим глазом. Она состоит из компонента яркости (Lightness) и двух цветовых осей: A (от зелёного к красному) и B (от синего к жёлтому). LAB не зависит от конкретного устройства, что делает её полезной при преобразовании изображений между различными цветовыми пространствами и для точной цветокоррекции. Данная цветовая модель охватывает более широкий спектр цветов, чем RGB или CMYK.

5. **HSB (Hue, Saturation, Brightness) [HSB]** — цветовая модель, описывающая цвет с точки зрения оттенка, насыщенности и яркости. Основной цвет определяется оттенком, насыщенность показывает его чистоту и интенсивность, а яркость отражает степень светлоты. Благодаря своей интуитивности, HSB широко используется в приложениях для работы с цветом, таких как графические и дизайнерские программы.

1.4.2 Сравнение цветовых моделей изображений

Сравнение предлагается проводить по следующим критериям.

1. Класс метода по принципу действия.

2. Количество байт для кодирования одного пикселя.

3. Наличие поддержки альфа-канала.

4. Отсутствие отдельного канала для яркости.

Результаты сравнения цветовых моделей изображений приведены в таблице **??**.

Таблица 1.3 – Сравнение рассмотренных цветовых моделей изображений

Цветовая модель	**Кр. 1**	**Кр. 2**	**Кр. 3**	**Кр. 4**
RGB	аддитивный	3	–	+
RGBA	аддитивный	4	+	+
CMYK	субтрактивный	4	–	+
LAB	перцепционный	3	–	–
HSB	перцепционный	3	–	–

1.5 Постановка задачи

В рамках выполнения выпускной квалификационной работы требуется разработать метод сжатия статических изображений без потерь на основе алгоритма Хаффмана. При создании метода необходимо определить:

— входные и выходные данные метода сжатия;

— способ хранения информации, необходимой для восстановления исходного качества изображений.

В разрабатываемом гибридном методе сжатия улучшение будет производиться за счет первичной обработки изображения другим методом сжатия, а именно словарным методом LZW. Формализованная постановка задачи в виде IDEF0-диаграммы представлена на рисунке **??**.

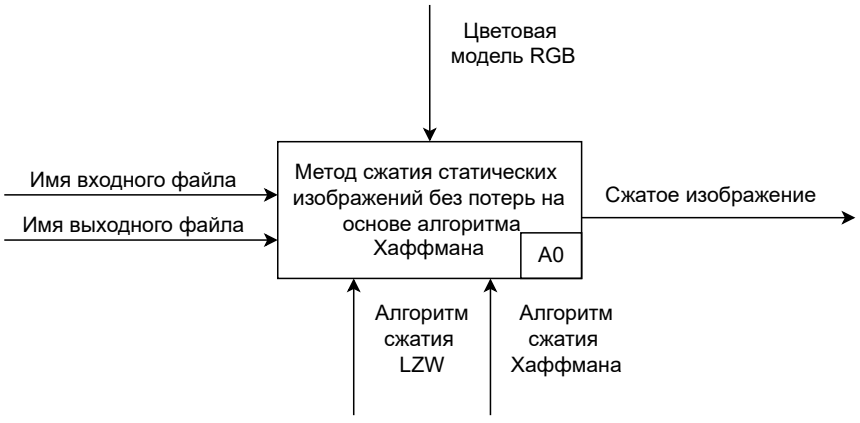

Рисунок 1.2 – Формализованная постановка задачи в нотации IDEF0

Вывод

В данном разделе была проведена классификация основных методов сжатия статических изображений по следующим категориям: сжатие с потерями и сжатие без потерь. В каждой из них было проведено сравнение описанных методов по выделенным критериям. Также были рассмотрены основные цветовые модели изображений.

Унарное кодирование может применяться для сжатия изображений в тех случаях, когда значения пикселей имеют ограниченный диапазон и малую вариацию (черно-белые снимки). Алгоритмы RLE и LZW могут быть полезны для изображений с большими областями одного цвета или повторяющихся участков. Арифметическое кодирование следует использовать для изображений с большим количеством текста, например, для отсканированных документов, где вероятностное распределение частоты появления символов может быть использовано в качестве основы для сжатия. Алгоритм Хаффмана подойдет для сжатия стандартных изображений без потерь, также он может использоваться внутри сжимающих форматов изображений, таких как JPEG для оптимизации размера файлов.

JPEG является одним из наиболее широко используемых методов сжатия изображений, и обычно используется для фотографий и непрерывных тональных изображений. Wavelet сжатие было специально разработано для цветных и черно-белых изображений с плавными переходами, из-за чего подходит для обработки рентгеновских снимков и МРТ. Изображения, представляющие природные сцены, такие как пейзажи, горы, облака, водопады, хорошо подходят для фрактального сжатия, поскольку обладают повторяющимися узорами и самоподобием.

2 Конструкторская часть

2.1 Требования к разрабатываемому методу сжатия изображений

Для гибридного метода сжатия изображений были выдвинуты следующие требования:

— на вход разрабатываемый метод должен получать путь до файла со статическим изображением и путь до директории, куда необходимо сохранить сжатый файл;

— результатом работы метода должно стать сжатое изображение, сохраненное в указанной директории;

— сжатое изображение должно содержать всю информацию, необходимую для его восстановления;

— сжатие должно сохранять всю информацию об исходном изображении.

2.2 Проектирование метода сжатия изображений

Разрабатываемый метод сжатия статических изображений будет представлять собой гибридный метод на основе алгоритма сжатия Хаффмана. Улучшение данного метода будет производиться за счет первичной обработки изображения другим алгоритмом сжатия [**my-article**]. В качестве такого метода был выбран LZW из-за:

— возможности кодирования данных за один проход;

— отсутствия необходимости в таблице частот пикселей сжимаемого изображения.

Метод LZW удаляет избыточность из последовательности пикселей изображения. Он заменяет повторяющиеся подстроки уникальными кодами, что значительно уменьшает размер изображения и приводит к уменьшению размера дерева кодов Хаффмана.

Разрабатываемый гибридной метод сжатия будет состоять из следующих этапов:

— получение данных сжимаемого изображения в виде байтовой строки, которая будет использоваться в качестве входных данных для LZW;

— выполнение первичного сжатия изображения алгоритмом LZW;

— нахождение таблицы частот символов;

— построение дерева кодов Хаффмана на основе вычисленной таблицы;

— выполнение повторного сжатия изображения алгоритмом Хаффмана;

— создание файла с сжатым изображением и информацией для его распаковки.

Таким образом, использование первичной обработки изображения в гибридном методе сжатия позволяет подготовить данные для метода Хаффмана путем уменьшения количества обрабатываемых символов. Такой подход приводит к более эффективному сжатию Хаффмана и, следовательно, к более высокой общей степени сжатия.

Диаграмма уровня A1 (рисунок **??**) иллюстрирует общую структуру гибридного метода сжатия: преобразование изображения в байтовую строку, этап сжатия и создание итогового файла с сжатым изображением.

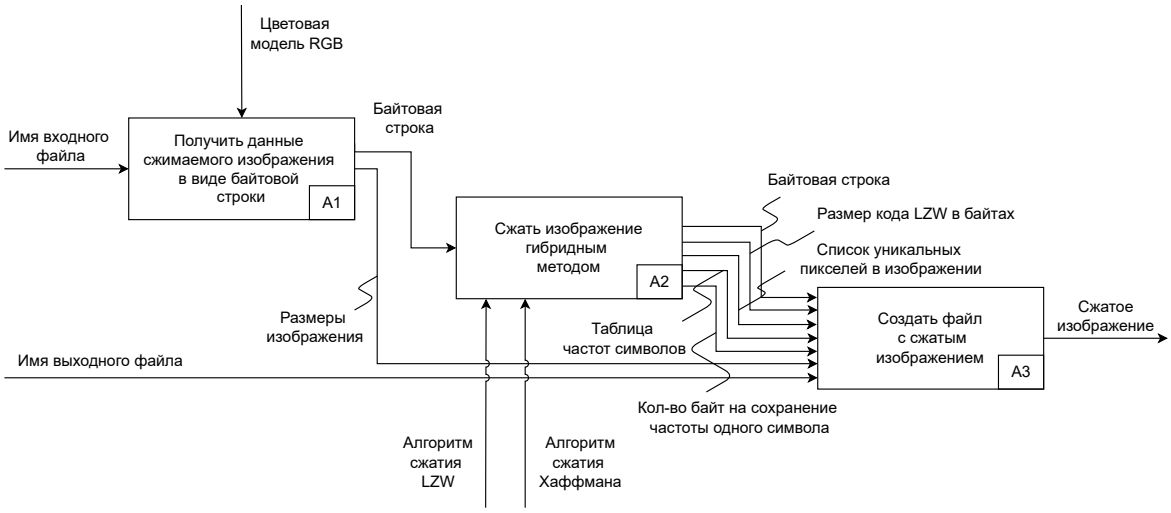

Рисунок 2.1 – Детализированная IDEF0-диаграмма гибридного метода сжатия изображений первого уровня

Диаграмма уровня A2 (рисунок ??) раскрывает детали этапа сжатия, выделяя первичную обработку изображения методом LZW, построение таблицы частот символов, генерацию дерева Хаффмана и повторное кодирование методом Хаффмана.

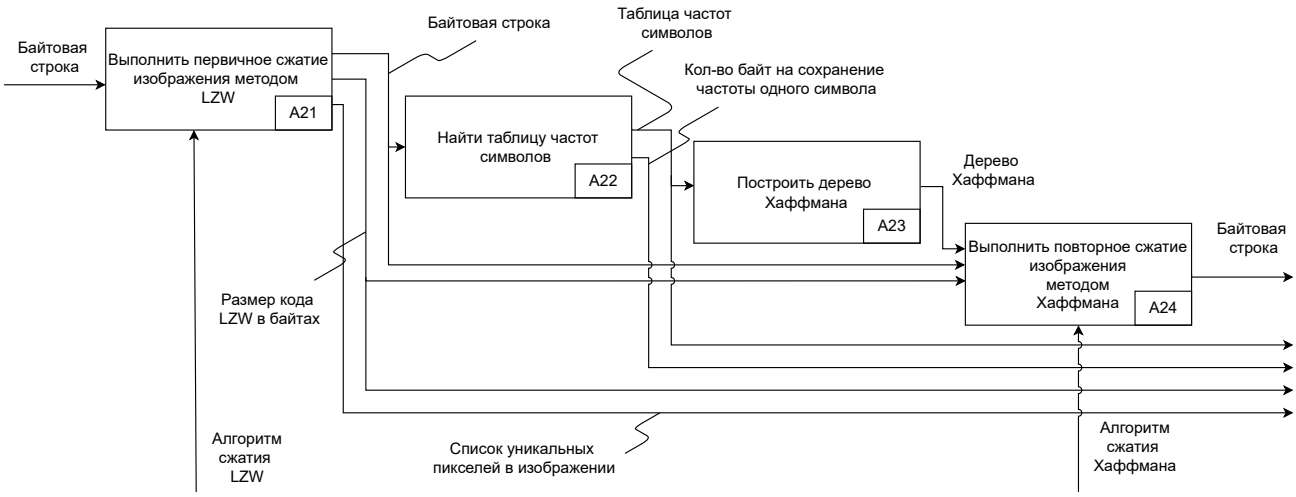

Рисунок 2.2 – Детализированная IDEF0-диаграмма гибридного метода сжатия изображений уровня A2

2.3 Требования к разрабатываемому ПО

Для демонстрации работы гибридного метода необходимо разработать ПО со следующими требованиями:

— взаимодействие пользователя с ПО должно осуществляться с помощью графического интерфейса;

— необходимо предусмотреть возможность выбора сжимаемого изображения через файловый менеджер;

— необходимо предусмотреть возможность восстановления сжатых изображений;

— пользователь должен иметь возможность сравнения гибридного метода сжатия изображений с базовыми, на основе которых он был разработан;

— сравнение должно проводиться по степени сжатия исходного файла, а также по размеру информации для распаковки в сжатом изображении.

Также необходимо подготовить список тестовых изображений для сжатия и положить их в директорию `input_data`.

2.4 Схемы разрабатываемого гибридного метода сжатия изображений

2.4.1 Схема гибридного метода сжатия

Схема гибридного метода сжатия статических изображений представлена на рисунке ??. Она состоит из шести основных пунктов, три из которых далее будут рассмотрены подробно.

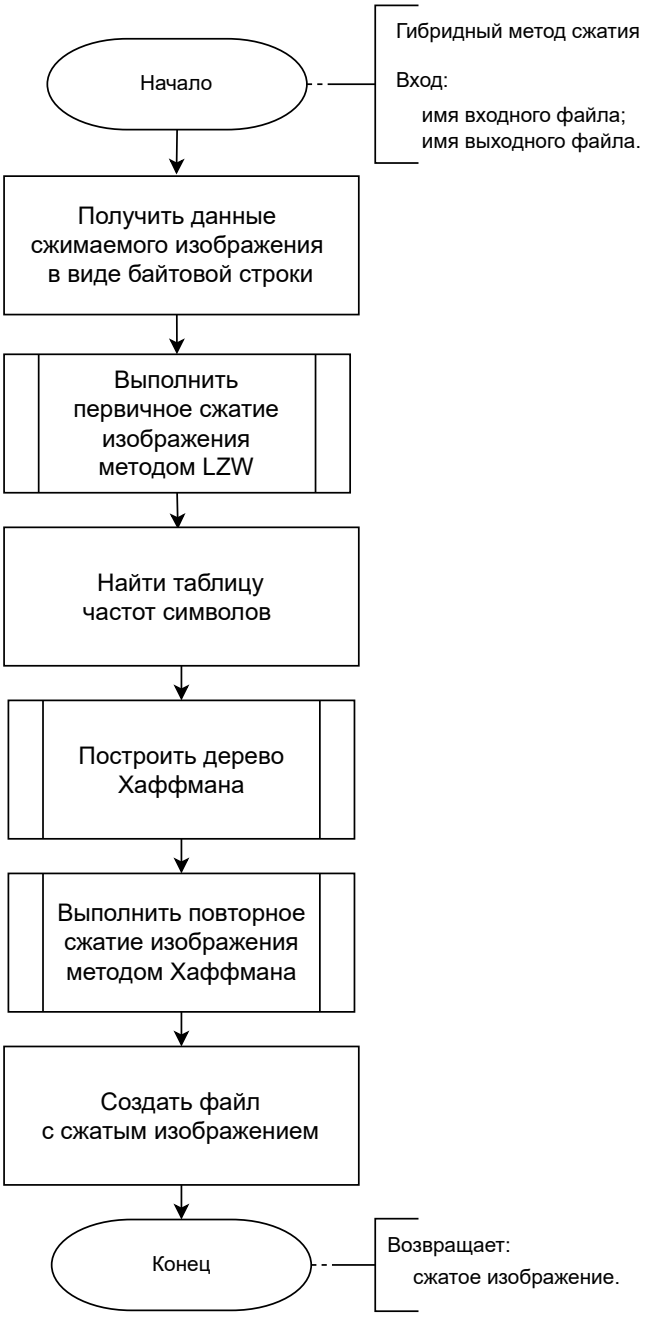

Рисунок 2.3 – Схема гибридного метода сжатия изображений

2.4.2 Схема метода LZW для первичного сжатия данных

Схема метода первичного сжатия LZW представлена на рисунке ??. На данном этапе происходит удаление избыточности данных и уменьшение количества обрабатываемых символов.

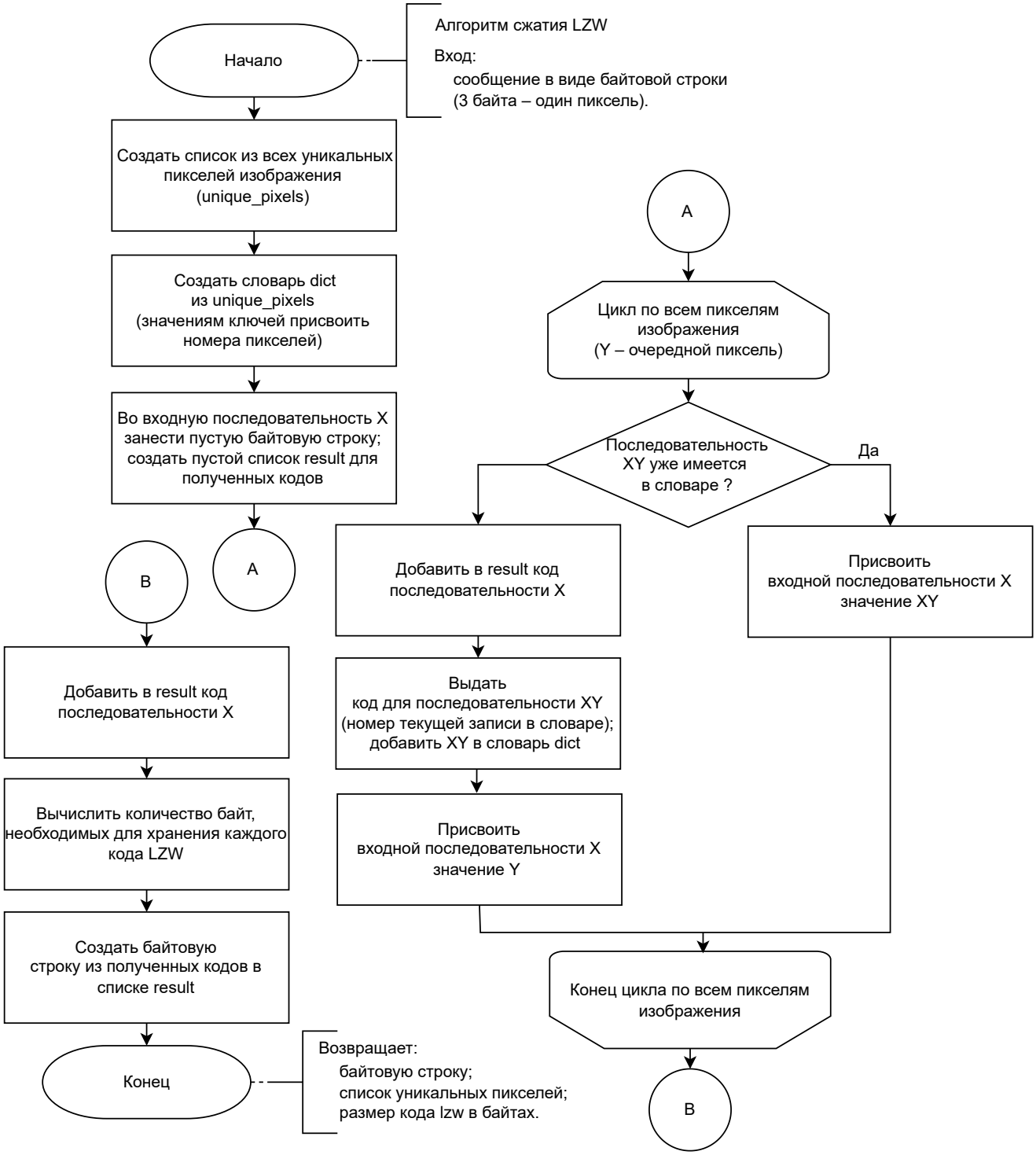

Рисунок 2.4 – Схема метода LZW для первичного сжатия изображений

2.4.3 Схема построения дерева кодов Хаффмана

Схема построения дерева кодов Хаффмана на основе таблицы частот символов представлена на рисунке **??**. На основе этого дерева будет поизведено сжатие байтовой строки, полученной на этапе первичного сжатия изображения методом LZW.

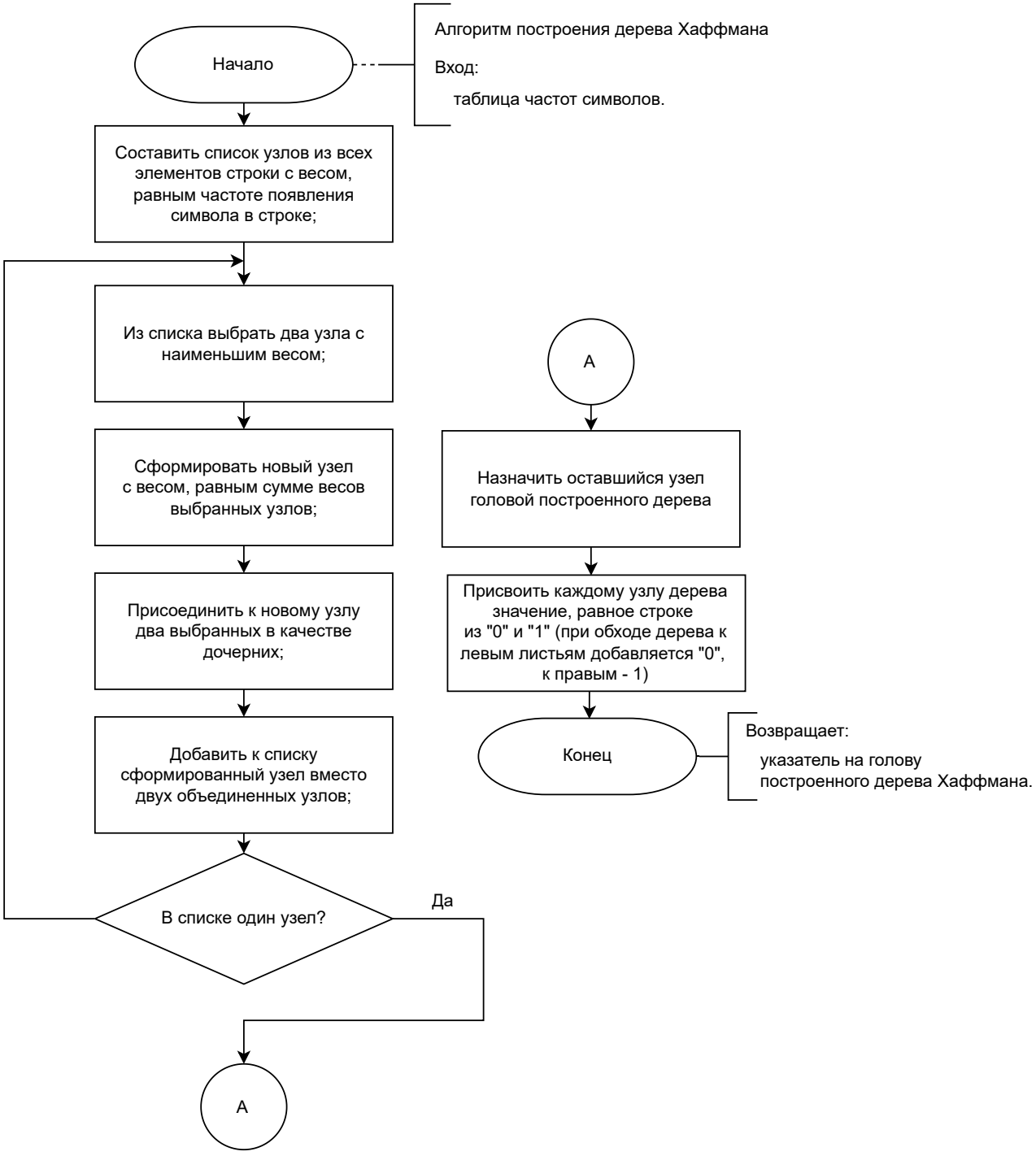

Рисунок 2.5 – Схема построения дерева кодов Хаффмана

2.4.4 Схема метода Хаффмана для повторного сжатия данных

Схема метода Хаффмана для повторного сжатия данных представлена на рисунке **??**. Это основной этап гибридного метода, в результате которого будет получена байтовая строка с итоговым сжатым изображением.

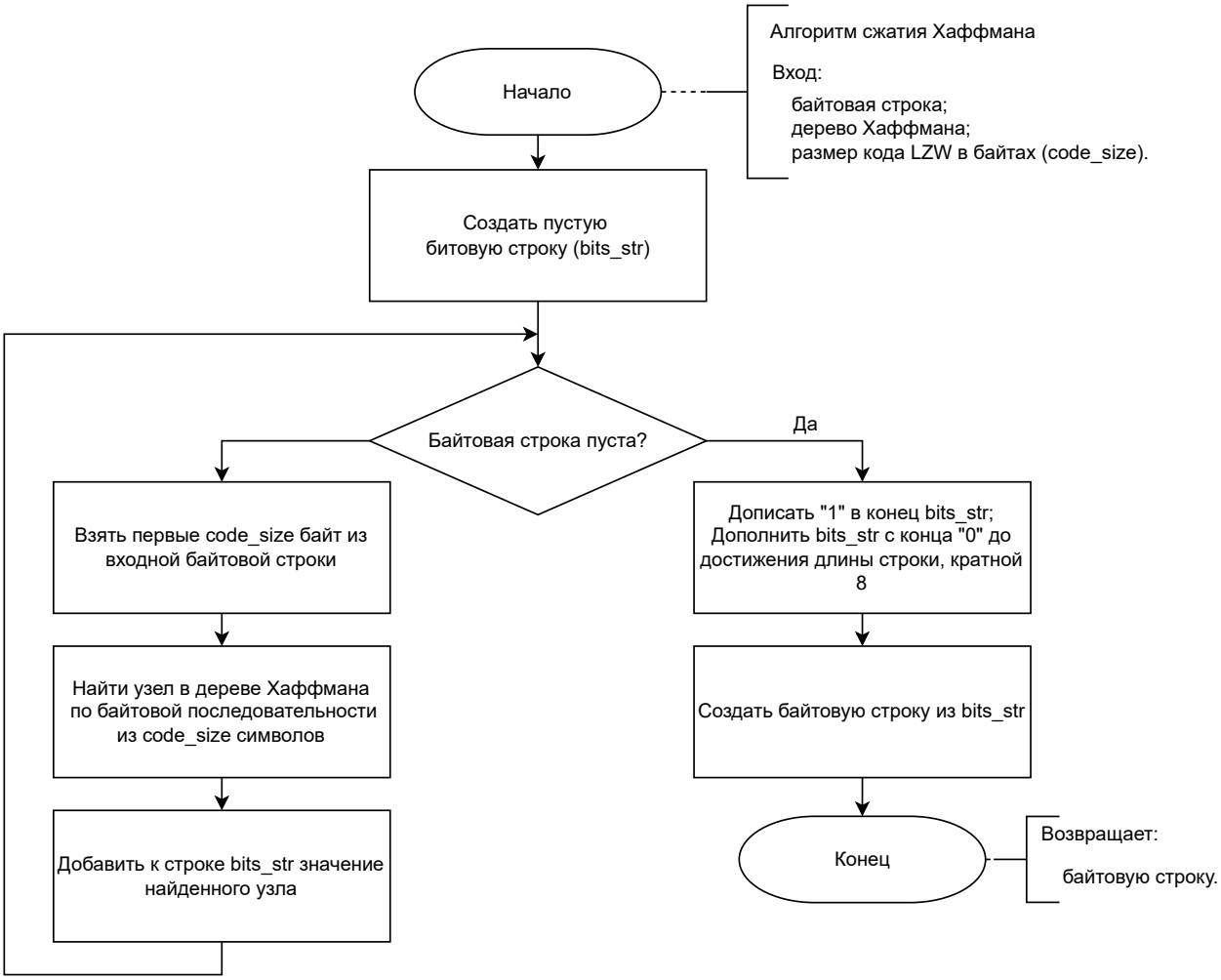

Рисунок 2.6 – Схема метода Хаффмана для повторного сжатия данных

Вывод

В данном разделе были предъявлены требования к разрабатываемому методу сжатия статических изображений и к разрабатываемому ПО, произведено проектирование метода сжатия. Для первичного сжатия, удаления избыточности и уменьшения количества обрабатываемых символов был выбран метод LZW. Кроме того, в данном разделе были построены схемы для реализации гибридного метода сжатия.

3 Технологическая часть

3.1 Используемые программные средства для реализации метода

В качестве языка программирования был выбран `Python` [**Python**]. Для `Python` существует большое количество библиотек и документация на русском языке, а сам язык поддерживает объектно-ориентированную парадигму программирования.

При создании графического интерфейса для программного обеспечения была использована библиотека `tkinter` [**Tkinter**]. Она является кроссплатформенной и включена в стандартную библиотеку языка `Python` в виде отдельного модуля. Для визуализации сравнения работы методов сжатия изображений использовалась библиотека `matplotlib` [**Matplotlib**] с следующими модулями:

— `matplotlib.pyplot` [**Pyplot**] — модуль, предоставляющий функции для создания графиков и визуализации данных, использовался для построения столбчатых диаграмм при сравнении методов сжатия изображений;

— `matplotlib.offsetbox` [**OffsetBox**] — модуль, предоставляющий возможность размещения текстовых и графических элементов на построенных графиках, использовался для добавления сжимаемых изображений под диаграммами сравнения методов.

Для работы с массивами битов при сжатии данных методом Хаффмана была использована библиотека `bitarray` [**Bitarray**], для отображения прогресса этапов сжатия и распаковки изображения использовалась библиотека `progress` [**Progress**]. Для получения списка файлов, доступных для сжатия, был использован модуль `subprocess` [**Subprocess**].

3.2 Формат входных и выходных данных

В качестве входных данных разработанный программный комплекс получает путь до изображения в формате BMP, TIFF, PNG или JPEG, а также путь до директории, куда необходимо сохранить сжатое и распакованное изображения. Также пользователю предоставляется возможность выбрать один из

трех методов сжатия: LZW, Хаффман или гибридный метод, разработанный в данной работе.

На выходе в директории с результатами создаются два файла с расширениями .bin (для сжатого изображения) и .bmp (для распакованного). В консоль выводится подробная информация об этапах сжатия и распаковки изображения, размеры исходного и полученного файлов, а также итоговая степень сжатия.

3.3 Структура разработанного ПО

3.3.1 Описание этапов гибридного метода сжатия

Реализация гибридного метода сжатия статических изображений без потерь состоит из следующих основных этапов.

1. Первичное сжатие изображения методом lZW.
2. Создание таблицы частот символов.
3. Построение дерева Хаффмана.
4. Применение метода сжатия Хаффмана к байтовой строке, полученной после первого этапа алгоритма.
5. Создание файла с сжатым изображением и информацией для его распаковки.

На первом этапе метода проводится первичное сжатие изображения методом LZW. В процессе обработки пикселей входного изображения создается словарь повторяющихся цепочек байт. Выделенные последовательности пикселей заменяются на уникальные коды фиксированной длины. Размер таких кодов зависит от количества заменяемых последовательностей (чем длиннее код, чем больше цепочек байт можно заменить на первом этапе метода). При распаковки сжатого изображения используется тот же словарь повторяющихся цепочек пикселей.

На втором этапе подсчитывается количество каждого уникального кода в полученной байтовой строке, строится таблица частот символов.

Третий этап включает в себя построение дерева Хаффмана, задача которого заключается в присвоении часто встречающимся символам более

коротких кодов, а редко встречающимся — более длинных. В отличии от классического дерева Хаффмана, в разработанном методе за один символ принимается не один байт, а уникальный код, состоящий из заданного числа байт.

На четвертом этапе происходит применение метода сжатия Хаффмана к байтовой строке, полученной после первого этапа алгоритма. На основе построенного дерева каждой цепочке байт (уникальному коду из метода LZW) присваивается код Хаффмана переменной длины, который за счет уникального префикса может быть однозначно декодирован.

На заключительном этапе метода происходит формирование байтовой строки со сжатым изображением и информацией для его распаковки, которая включает в себя таблицу частот символов (для восстановления дерева Хаффмана) и уникальные пиксели исходного изображения (для воссоздания словаря повторяющихся цепочек байт). Полученная байтовая строка является результатом сжатия статического изображения разработанным гибридным методом.

3.3.2 Описание модулей разработанного ПО

UML-диаграмма [**UML**] компонентов разработанного программного обеспечения представлена на рисунке **??**. Она показывает структуру зависимостей между основными модулями программы.

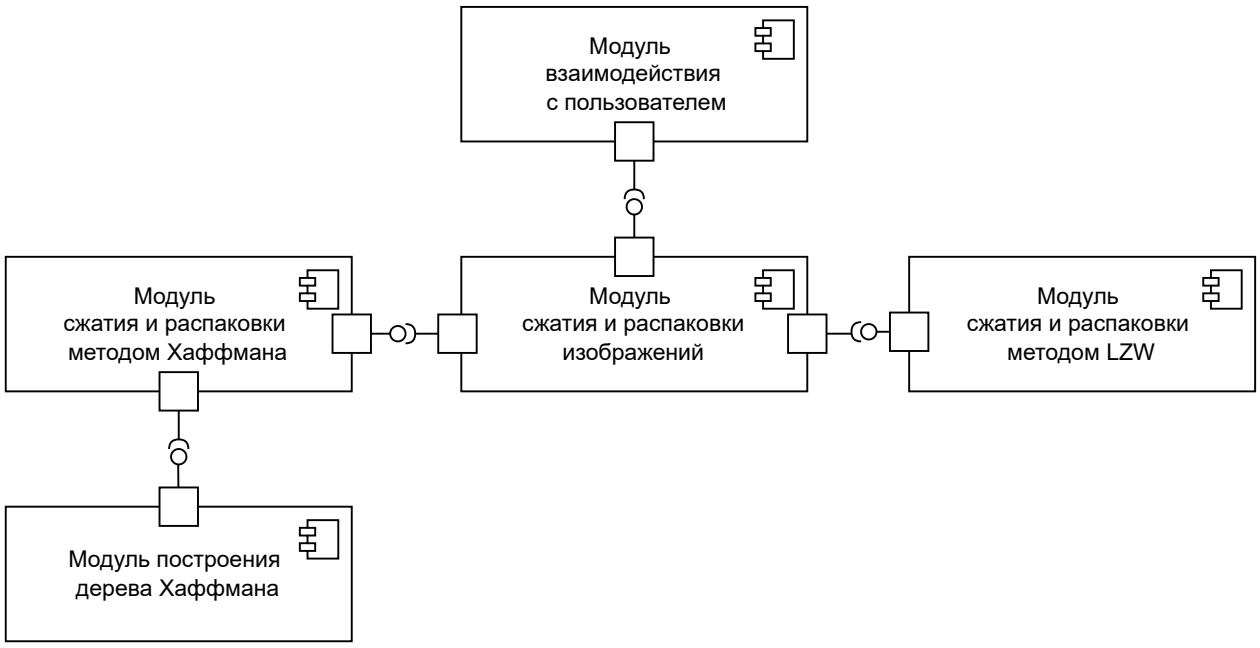

Рисунок 3.1 – UML-диаграмма компонентов разработанного ПО

Разработанное программное обеспечение состоит из следующих модулей.

1. **Модуль сжатия и распаковки изображений**. Реализует основной функционал сжатия и распаковки изображений разработанным гибридным методом (листинг **??**). Основной класс модуля, `Compression`, отвечает за выполнение всех этапов сжатия и восстановления данных. При сжатии входное изображение представляется в виде байтовой строки, к которой применяется выбранный метод (LZW, Хаффман или гибридный), после чего формируется файл с сжатым изображением и метаданными для его распаковки.

2. **Модуль сжатия и распаковки методом LZW**. Реализует алгоритм сжатия и распаковки данных методом LZW (листинг **??**). Включает создание словаря повторяющихся последовательностей байт и их замену уникальными кодами фиксированной длины. При сжатии генерирует сжатую байтовую строку и список уникальных пикселей, а при распаковке восстанавливает исходные данные на основе словаря, воссозданного по списку пикселей.

3. **Модуль сжатия и распаковки методом Хаффмана**. Реализует алгоритм сжатия и распаковки данных методом Хаффмана (листинг **??**). Включает построение таблицы частот символов, создание дерева Хаффмана и генерацию кодов переменной длины. При сжатии преобразует данные в битовую строку на основе построенного дерева, а при распаковке восстанавливает исходные данные на основе сохраненной таблице частот символов.

4. **Модуль построения дерева Хаффмана**. Предоставляет вспомогательные классы и функции для работы с деревом Хаффмана (листинг **??**). Включает создание узлов дерева, объединение их на основе частот символов и генерацию кодов Хаффмана. Модуль используется в `huffman.py` для построения дерева и кодирования данных, а также для восстановления исходной информации при распаковке.

5. **Модуль взаимодействия с пользователем**. Реализует графический интерфейс (листинг **??**) с использованием библиотеки `tkinter`. Позволяет пользователю выбрать входное изображение, метод сжатия (LZW,

Хаффман или гибридный), а также директорию для сохранения результатов. Модуль отображает прогресс выполнения операций, результаты сжатия и распаковки, а также предоставляет визуализацию сравнения методов сжатия.

3.4 Результаты работы ПО

Разработанное программное обеспечение представляет собой приложение с графическим интерфейсом (рисунок **??**), предоставляющее возможность выбора исходного изображения, метода сжатия и директории для сохранения результатов. Пользователь может сжать и распаковать выбранное изображение, посмотреть результаты сравнения доступных методов сжатия, а также получить информацию о данной программе. Подробная информация об этапах сжатия и распаковки выводится как в консоль, так и в окно программы.

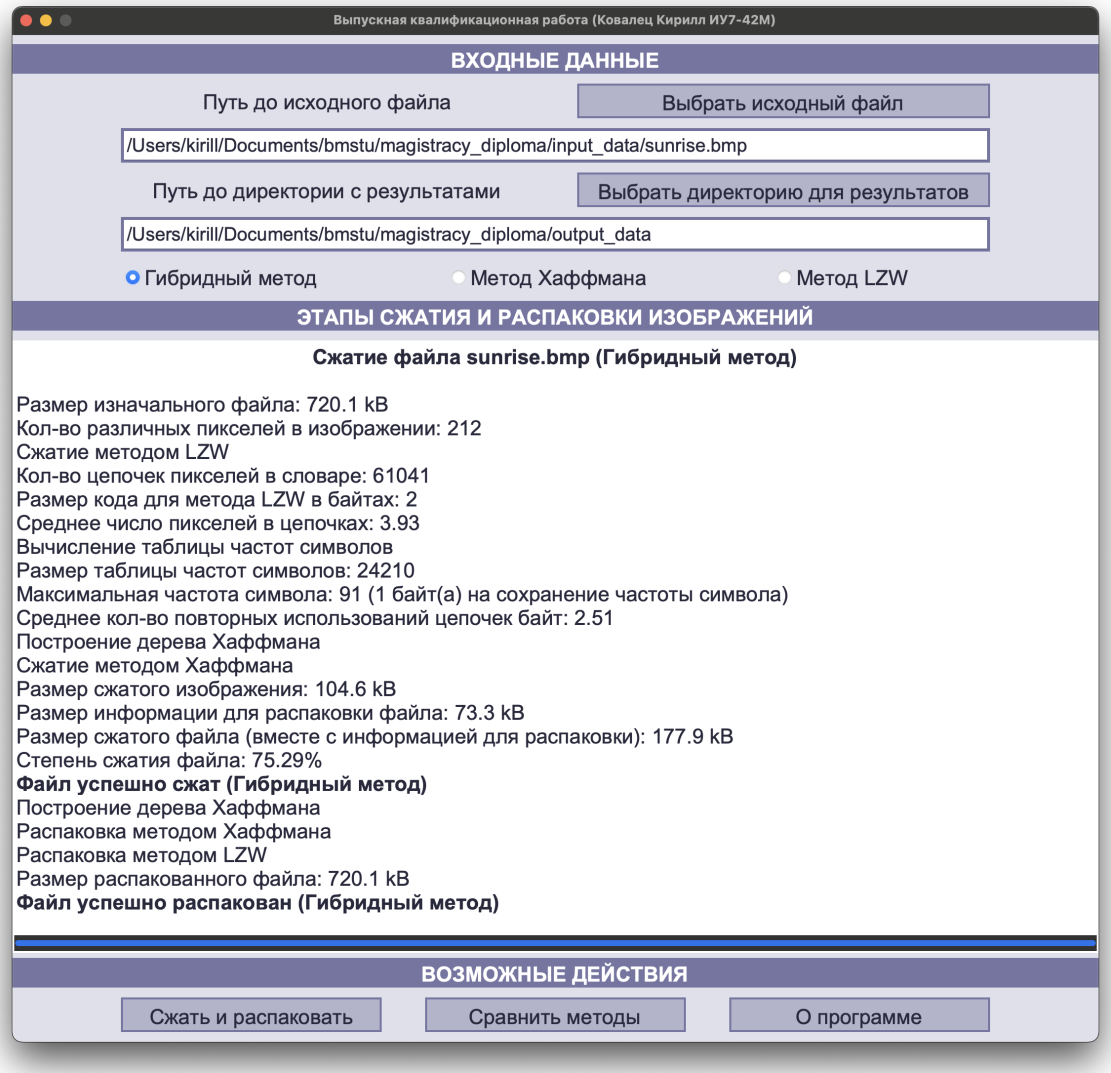

Рисунок 3.2 – Интерфейс программы для сжатия изображений

Для демонстрации работы гибридного метода сжатия было выбрано изображение `sunrise.bmp`, представленное на рисунке **??**.

Рисунок 3.3 – Исходное изображение восхода солнца

После сжатия файла пользователю выводится график сравнения размеров полученного изображения с исходным в виде столбчатой диаграммы (рисунок **??**). Данный график позволяет оценить коэффициент сжатия файла и размер методанных, необходимых для распаковки изображения.

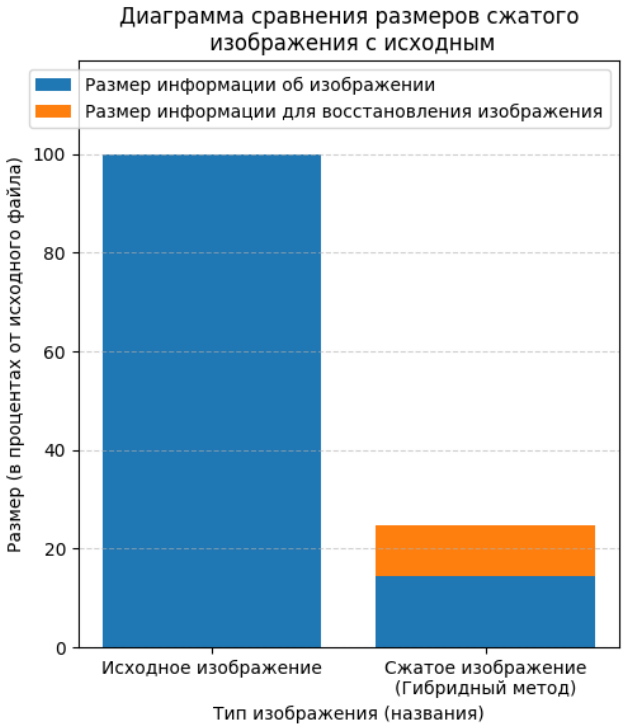

Рисунок 3.4 – Результаты сравнения размеров сжатого изображения с исходным (изображение восхода солнца)

Подробная информация об этапах сжатия и распаковки изображения восхода солнца продемонстрирована в листинге ??.

Листинг 3.1 — Результаты сжатия и распаковки входного изображения восхода солнца гибридным методом

Распакованное изображение `sunrise.bmp` с сохранением всех деталей исходного представлено на рисунке ??.

Рисунок 3.5 – Восстановленное изображение восхода солнца

Для изображения `sunrise.bmp` исходный размер файла составил 720.1 КБ, а количество различных пикселей в изображении — 212. После сжатия разработанным гибридным методом (LZW + Хаффман) размер сжатого изображения составил 104.6 КБ, а информация для его распаковки заняла 73.3 КБ, что в сумме дало размер сжатого файла 177.9 КБ. Степень сжатия файла составила 75.29%.

Среднее число пикселей в цепочках, созданных методом LZW, составило 3.93, а среднее количество повторных использований цепочек байт — 2.51. Распаковка файла успешно восстановила исходное изображение с размером 720.1 КБ.

Для следующего примера работы гибридного метода сжатия было создано изображение `girl.bmp`, представленное на рисунке ??.

Рисунок 3.6 – Исходное изображение девушки

Сравнение размеров сжатого файла с исходным для изображения `girl.bmp` представлено на рисунке ??.

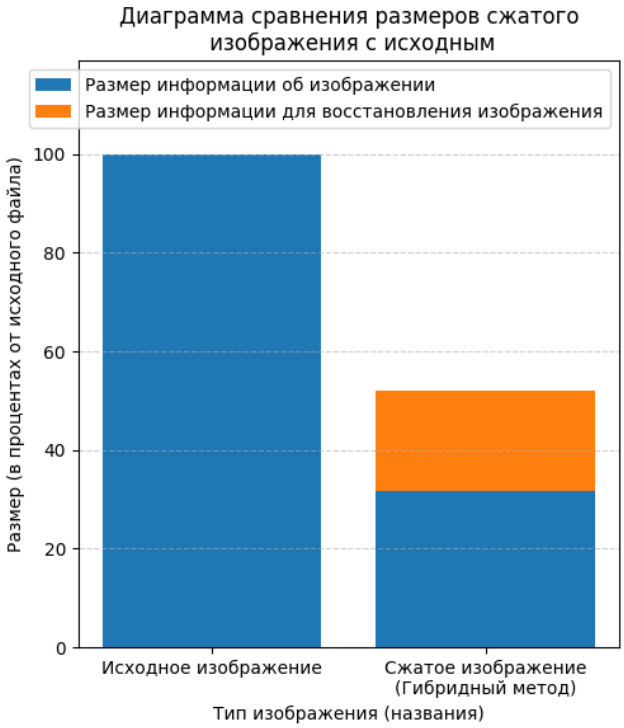

Рисунок 3.7 – Результаты сравнения размеров сжатого изображения с исходным (изображение девушки)

Подробная информация об этапах сжатия и распаковки изображения девушки продемонстрирована в листинге ??.

Листинг 3.2 — Результаты сжатия и распаковки входного изображения девушки гибридным методом

Распакованное изображение `girl.bmp` с сохранением всех деталей исходного представлено на рисунке ??.

Рисунок 3.8 – Восстановленное изображение девушки

Для изображения `girl.bmp` исходный размер файла составил 491.3 КБ, а количество уникальных пикселей в изображении — 2643. После применения гибридного метода сжатия (LZW + Хаффман) размер сжатого изображения составил 155.5 КБ, а данные для его восстановления заняли 99.7 КБ, что в сумме дало общий размер сжатого файла 255.3 КБ. Степень сжатия составила 48.05%.

Средняя длина цепочек пикселей, сформированных методом LZW, составила 1.69, а среднее количество повторений цепочек байт — 4.11. Размер таблицы частот символов, использованной для построения дерева Хаффмана, составил 22.9 КБ. Максимальная частота символа в таблице составила 98, что потребовало 1 байт для хранения частоты. Распаковка файла успешно восстановила исходное качество изображения.

Вывод

В данном разделе были рассмотрены используемые программные средства реализации метода, описан формат входных и выходных данных, описана реализация гибридного метода сжатия статических изображений и приведены результаты работы программы. Также было представлено описание структуры разработанного ПО.

В примере изображения с восходом солнца степень сжатия в 1.56 раз больше, чем в изображении с девушкой (75.29% против 48.05%). Это связано с тем, что в файле `sunrise.bmp` цепочки байт, полученные на этапе обработки изображения методом LZW, в среднем содержат больше пикселей на 133% (3.93 против 1.69). Также в изображении `girl.bmp` больше уникальных пикселей (2643 против 212, то есть в 12.47 раз). Эти факторы способствуют более эффективному сжатию изображения с восходом солнца разработанным гибридным методом.

4 Исследовательская часть

4.1 Критерии оценки методов сжатия изображений

Для оценки методов сжатия изображений использовались следующие критерии.

1. **Степень сжатия**: показывает, на сколько процентов от изначального размера файла удалось сжать изображение. Чем выше коэффициент, тем лучше удалось выполнить сжатие. При этом учитывается не только размер сжатого изображения, но и объем метаданных, необходимых для его восстановления. Степень сжатия рассчитывается по формуле.

$$\text{Степень сжатия} = \left(1 - \frac{\text{Размер сжатого изображения}}{\text{Размер исходного изображения}}\right) \times 100\%. \tag{4.1}$$

2. **Размер информации для распаковки**: показывает, какую часть сжатого изображения занимает информация, необходимая для восстановления исходного файла. Чем выше этот показатель, тем большую долю от сжатого файла занимают метаданные. Большой объем информации для распаковки может не дать достичь высокой степени сжатия изображения.

Для проведения исследования по выделенным критериям были выбраны изображения в формате BMP [**article-bmp**]. Выбор файлов данного формата обусловлен следующими причинами:

— Файлы в формате BMP хранят информацию о каждом пикселе изображения в исходном качестве без сжатия.

— BMP-файлы широко используются на практике в различных приложениях и системах.

— Формат BMP подходит для работы как с черно-белыми, так и с цветными изображениями.

4.2 Сравнение разработанного метода сжатия с аналогами

4.2.1 Сравнение по степени сжатия изображений

Результаты сравнения методов сжатия статических изображений без потерь по степени сжатия приведены в таблице ?? и продемонстрированы на рисунке ??.

Таблица 4.1 – Результаты сравнения методов сжатия изображений по степени сжатия

Изображение	**Метод LZW, %**	**Гибридный метод, %**	**Метод Хаффмана, %**
sunrise.bmp	83.01	75.29	69.91
mars.bmp	84.61	78.30	71.59
wheat.bmp	77.67	70.52	68.02
forest.bmp	44.52	54.23	67.77
girl.bmp	40.84	48.05	59.23

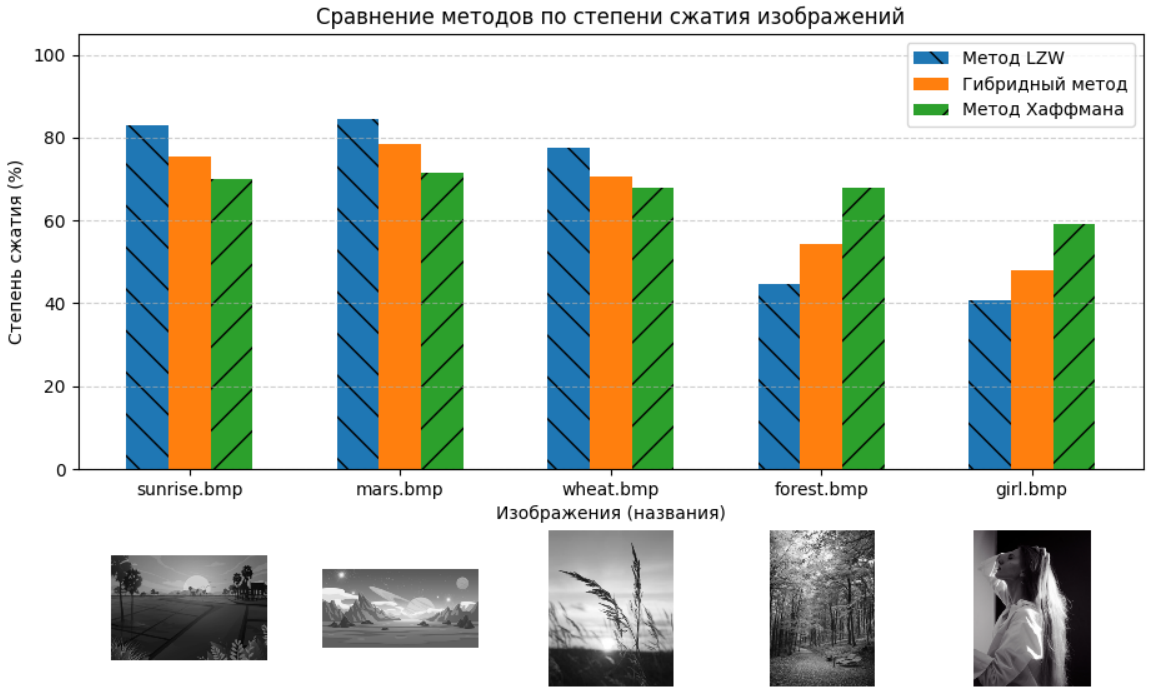

Рисунок 4.1 – Сравнения методов сжатия статических изображений без потерь по степени сжатия

На примерах видно, что степень сжатия зависит от типа данных: метод LZW лучше работает с длинными последовательностями одинаковых пикселей, показывая наивысшую степень сжатия для изображений *sunrise.bmp* (83.01%), *mars.bmp* (84.61%) и *wheat.bmp* (77.67%). Метод Хаффмана, напротив, демонстрирует лучшие результаты для изображений с неравномерным распределением цветов, таких как *forest.bmp* (67.77%) и *girl.bmp* (59.23%).

Гибридный метод является универсальным решением, которое позволяет минимизировать зависимость от особенностей входных изображений. Например, для изображения *forest.bmp* он показывает степень сжатия 54.23%, что выше, чем у метода LZW (44.52%), но ниже, чем у метода Хаффмана (67.77%). В то же время для изображения *mars.bmp* его результат (78.30%) уступает методу LZW (84.61%), но превосходит метод Хаффмана (71.59%).

Таким образом, гибридный метод обеспечивает более стабильный результат сжатия, выступая как компромиссное решение между высокой степенью сжатия и универсальностью.

4.2.2 Сравнение по размеру информации для распаковки изображений

Результаты сравнения методов сжатия статических изображений без потерь по количеству информации, необходимой для распаковки изображений, приведены в таблице **??**.

Таблица 4.2 – Результаты сравнения методов сжатия по размеру информации для распаковки изображений

Изображение	**Метод LZW, %**	**Гибридный метод, %**	**Метод Хаффмана, %**
sunrise.bmp	0.53	41.20	0.51
mars.bmp	0.77	41.81	0.72
wheat.bmp	0.76	39.07	0.88
forest.bmp	0.25	33.50	0.73
girl.bmp	2.72	39.05	6.59

Визуализация результатов сравнения приведена на рисунке ??.

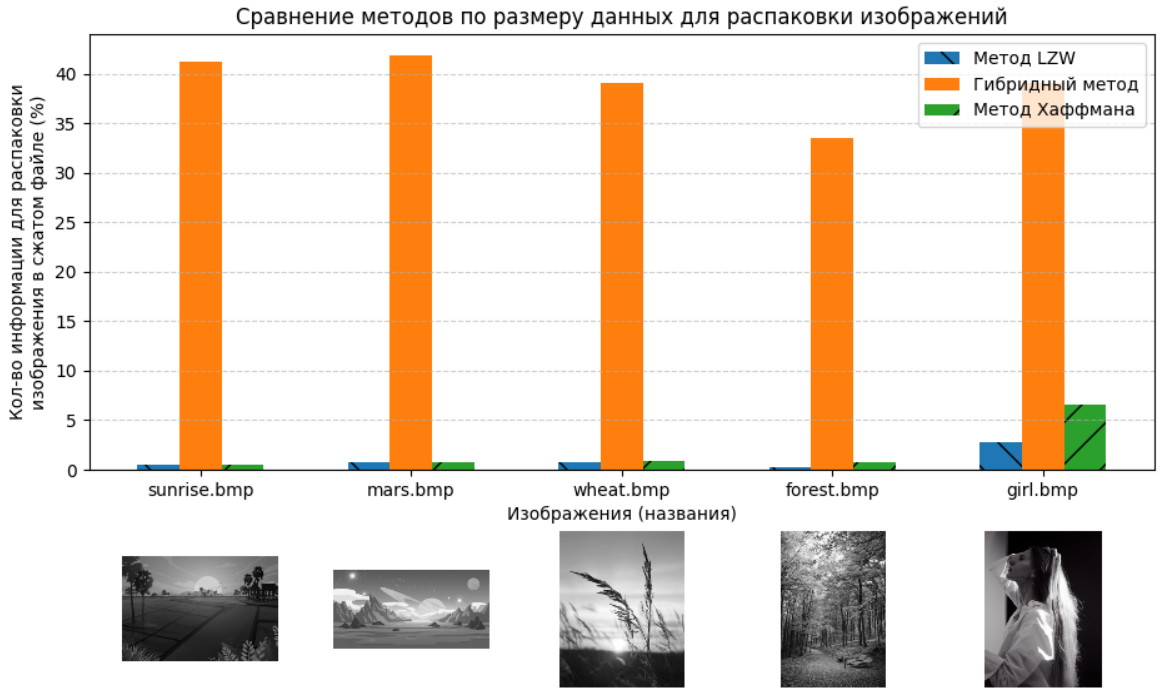

Рисунок 4.2 – Сравнение методов сжатия статических изображений без потерь по количеству информации для распаковки

Гибридный метод сжатия, сочетающий в себе алгоритмы LZW и Хаффмана, требует больше информации для распаковки изображения по сравнению с отдельными методами. Это связано с необходимостью сохранения данных, используемых на обоих этапах сжатия. Например, после применения LZW сохраняются уникальные пиксели изображения и размер кода LZW, а на этапе Хаффмана добавляются таблица частот символов и количество байт, необходимых для сохранения частоты одного символа. Каждый из этих компонентов увеличивает объем метаданных. Как видно из таблицы ??, для гибридного метода доля информации для распаковки составляет от 33.50% до 41.81%, тогда как для LZW и Хаффмана этот показатель не превышает 6.59%.

Выводы

Несмотря на больший объем метаданных в сжатом файле, гибридный метод обеспечивает более стабильный результат сжатия за счет сильных сторон обоих алгоритмов. LZW эффективно сжимает повторяющиеся последовательности (например, однородные участки изображений), а Хаффман оптимизирует кодирование частых символов. Это позволяет гибридному мето-

ду адаптироваться к разным типам изображений, минимизируя зависимость от их структуры. Например, для изображения forest.bmp гибридный метод показывает степень сжатия 54.23%, что лучше, чем у LZW (44.52%), и близко к результату Хаффмана (67.77%). Для изображения mars.bmp метод LZW демонстрирует степень сжатия 84.61%, что значительно лучше, чем у метода Хаффмана (71.59%). Гибридный метод показывает промежуточный результат (78.30%), сохраняя универсальность. Таким образом, компромисс между объемом метаданных и универсальностью делает гибридный метод подходящим для задач, где важна стабильность, а не абсолютная минимизация размера файла.

ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной квалификационной работы разработан метод сжатия статических изображений без потерь на основе алгоритма Хаффмана. Поставленная цель была достигнута.

Проведен аналитический обзор известных методов сжатия статических изображений. Были рассмотрены алгоритмы сжатия без потерь (RLE, LZW, унарное кодирование, Хаффман и арифметическое кодирование) и с потерями (JPEG, Wavelet, фрактальное сжатие), проведено сравнение методов по выделенным критериям. Рассмотрены основные цветовые модели изображений (RGB, RGBA, CMYK, LAB, HSB).

Разработан метод сжатия статических изображений без потерь на основе алгоритма Хаффмана. Он представляет собой гибридную реализацию, где для первичной обработки изображения используется словарный метод LZW.

Разработано программное обеспечение для демонстрации работы созданного метода. Реализованная программа представляет собой графическое приложение, предоставляющее пользователю возможность выбора изображения для сжатия и распаковки.

Программное обеспечение позволяет провести сравнение гибридного метода с Хаффманом и LZW по степени сжатия изображений и размеру информации, необходимой для их распаковки. Гибридный метод показал более стабильный результат сжатия, минимизировав зависимость от особенностей входных изображений, несмотря на большую величину метаданных, необходимых для восстановления исходных файлов.

ПРИЛОЖЕНИЕ А

Реализация гибридного метода сжатия статических изображений без потерь

Листинг А.1 — Реализация модуля сжатия изображений

Листинг А.2 — Реализация модуля для построения дерева Хаффмана

Листинг А.3 — Реализация модуля сжатия методом Хаффмана

Листинг А.4 — Реализация модуля для сжатия методом LZW

Листинг А.5 — Реализация модуля взаимодействия с пользователем

ПРИЛОЖЕНИЕ Б

Реализация сравнения методов сжатия статических изображений без потерь

Листинг Б.1 — Модуль для сравнения методов сжатия

ПРИЛОЖЕНИЕ В