МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование)

01.03.02 Прикладная математика и информатика
(код и наименование направления подготовки, специальности)

Компьютерные технологии и математическое моделирование
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему «Разработка алгоритма сжатия для оперативной съемки и отправки проблемы для АХО ТГУ»

Обучающийся Е.Ф.Гавлонская
(И.О. Фамилия) (личная подпись)

Руководитель М.А.Тренина
(ученая степень, звание, И.О. Фамилия)

Консультант к.п.н., доцент О.Н. Брега

Тольятти 2023

Аннотация

Тема выпускной квалификационной работы: «Разработка алгоритма сжатия для оперативной съемки и отправки проблемы для АХО ТГУ».

Работа выполнена студенткой Тольяттинского государственного университета, института математики, физики и информационных технологий, группы ПМИб-1902б, Гавлонской Екатериной Федоровной.

Объект исследования: анализ систем разработки алгоритма сжатия для оперативной съемки и отправки проблемы для АХО ТГУ.

Структура бакалаврской работы представлена введением, тремя разделами, заключением и списком литературы.

Во введении описывается актуальность данной задачи, формулируется цель и ставятся задачи, которые необходимо решить.

В первом разделе исследуется предметная область и происходит сравнительный анализ алгоритмов сжатия изображения.

Во втором разделе описывается структура мобильного приложения, выбранный алгоритм для разработки модуля сжатия изображения и производится постановка задачи.

В третьем разделе производится реализация и тестирование разработанного мобильного приложения, а также демонстрируются результаты работы.

В заключении представлены выводы по проделанной работе.

В работе использована 1 таблица, 22 рисунка, список литературы содержит 25 литературных источников. Общий объем выпускной квалификационной работы составляет 57 страницы.

Abstract

The topic of the graduate qualification work: «Development of compression algorithm for operative problem shooting and sending for AHU TSU».

The work was done by the student of Togliatti State University, Institute of Mathematics, Physics and Information Technology, group PMIB-1902b, Gavlonskaya Ekaterina Fedorovna.

Object of the research: analysis of compression algorithm development systems for on-line shooting and sending problem for AHU TSU.

The structure of the bachelor's work is represented by an introduction

m, three sections, the conclusion and the list of references.

The introduction describes the relevance of the problem, formulates the goal and sets the tasks to be solved.

The first section explores the subject area and a comparative analysis of image compression algorithms.

The second section describes the structure of the mobile application, the selected algorithm for the development of the image compression module and the problem statement.

In the third section, the implementation and testing of the developed mobile application are done and the results of the work are demonstrated.

The conclusion presents the research results.

The work comprises 1 table, 22 pictures, the reference list contains 25 literature sources. The total volume of the graduate qualification work is 56 pages.

Содержание

Введение

С течением времени прогресс и информационные технологии стремительно развивается. На сегодняшний день, мобильные устройства и быстрый доступ к информации становятся все более популярными среди людей во всем мире. С каждым годом процент пользователей мобильных версий сайтов постоянно увеличивается, а компьютерные версии становятся все менее популярными. В связи с этим, корпоративные технологии должны постоянно развиваться, чтобы не потерять свою актуальность и востребованность. Особенно важными являются информационные системы для административно-хозяйственного обслуживания в ВУЗах.

Актуальность темы исследования заключается в том, что разработка алгоритма с возможностью отправки проблемы упростит процесс обращения заявителя к службе поддержки административно-хозяйственного обслуживания, а возможность оперативной съемки значительно сократит время, необходимое на получение информации.

Объектом исследования является алгоритм сжатия для оперативной съемки и отправки проблемы.

Предметом исследования является разработка алгоритма сжатия для оперативной съемки и отправки проблемы для АХО ТГУ.

Целью выпускной квалификационной работы является разработка алгоритма сжатия для оперативной съемки и отправки проблемы для АХО ТГУ.

Для достижения цели необходимо выполнить следующие задачи:

- исследовать предметную область;
- сравнить существующие алгоритмы сжатия изображения с потерей качества и без потери качества;
- определить основные инструменты и средства для разработки мобильного приложения;
- выбрать для реализации алгоритм сжатия изображения;

– спроектировать алгоритм мобильного приложения;
– разработать программную реализацию мобильного приложения;
– выполнить тестирование разработанного мобильного приложения;
– сформировать вывод о полученных результатах.

Данная работа содержит в себе введение, три раздела, заключение и список используемой литературы.

Во введении описывается актуальность данной задачи, определяется цель и устанавливаются задачи, которые необходимо решить.

В первом разделе исследуется предметная область и происходит сравнительный анализ алгоритмов сжатия изображения.

Во втором разделе описывается структура мобильного приложения, выбранный алгоритм для разработки модуля сжатия изображения и производится постановка задачи.

В третьем разделе производится реализация и тестирование разработанного мобильного приложения, а также демонстрируются результаты работы.

В заключении представлены итоги проделанной работы, а также выводы относительно достигнутых результатов.

1 Анализ существующих технологий

1.1 Исследование предметной области

Android – это операционная система для мобильных устройств, которая была разработана компанией Google на основе ядра Linux. Android предоставляет разработчикам широкий набор системных служб, таких как средства защиты, регулирование памятью, управление процессами и архитектуру драйверов, которые обеспечивают высокую степень безопасности и стабильности работы устройства. Это мощная платформа разработки, которая включает в себя все необходимое для создания современных приложений, в том числе графический интерфейс, мультимедийные возможности, поддержку многопоточности и многие другие функции.

Java – это многоплатформенный язык программирования, который используется для разработки объектно-ориентированных приложений. Он может быть использован не только для создания приложений, но и как самостоятельная платформа. Java является быстрым, безопасным и надежным языком программирования, который находит применение в различных областях, включая мобильные приложения, корпоративное ПО, приложения для работы с большими данными и серверные технологии.

Важной особенностью Java заключается в простоте его применения. Причины, по которым разработчики предпочитают Java другим языкам программирования это:

1. высококачественные учебные ресурсы. Для новых разработчиков доступно большое количество документаций, печатных материалов и курсов, которые помогаю на протяжении всего обучения;

2. встроенные функции и библиотеки. Java предоставляет разработчикам богатую экосистему встроенных функций и библиотек, что позволяет им избегать необходимости писать каждую функцию с нуля при разработке разнообразных приложений;

3. высококачественные инструменты разработки. Java предоставляет разработчикам высококачественные инструменты для автоматизированного редактирования, отладки, тестирования, развертывания и управления изменениями;

4. безопасность. Гибкая система безопасности позволяет виртуальной машине полностью контролировать исполнение программы.

На рисунке 1 показана платформа Android, которая представляет собой комплекс из множества различных компонентов, включая базовые приложения, набор программных интерфейсов (API) для управления внешним видом и поведением приложения, а также множество поддерживающих файлов и библиотек.

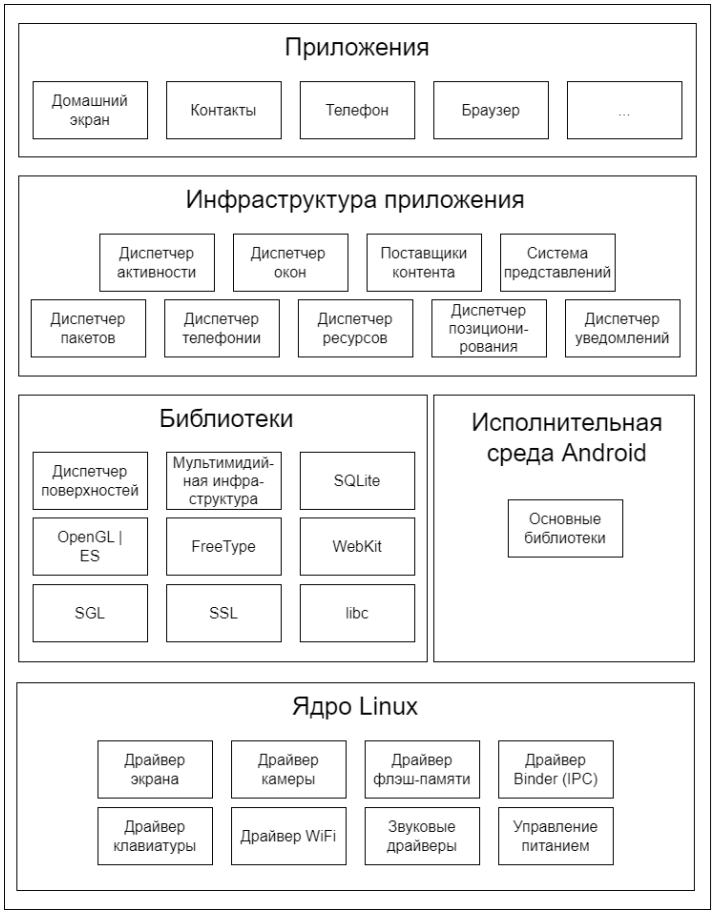

Рисунок 1 – Структура платформы Android

Android-приложение – это программное обеспечение, предназначенное

для работы на мобильных устройствах с операционной системой Android. Для построения Android-приложения следует выполнить следующие действия.

1. Подготовить среду разработки. Настройка Android Studio, включающая все необходимые компоненты для создания мобильных приложений под операционную систему Android.

Android Studio – специализированная версия IntelliJ IDEA, которая включает версию Android SDK и дополнительные инструменты графических интерфейсов, упрощающие разработку приложений.

IntelliJ IDEA – интегрированная система разработки программного обеспечения для различных языков программирования, разработанная компанией JetBrains.

2. Построение приложения. Создаем в Android Studio приложение, которое будет выводить текст на экран.

3. Запуск приложения в эмуляторе Android. Используем встроенный эмулятор, чтобы увидеть приложение в действии.

Android эмулятор является важным инструментом для разработчиков, который создается при помощи Android Virtual Device Manager (AVD Manager). Он обеспечивает полный доступ к Интернету, с возможность регулировать производительность и латентность соединения, а также имитировать входящие и исходящие телефонные вызовы и сообщения.

1.2 Алгоритмы сжатия изображений

Сжатие изображений представляет собой процесс использования алгоритмов сжатия данных для цифровых изображений, что позволяет уменьшить их размер и ускорить передачу изображения по сети, а также сэкономить место для хранения.

За последние десятилетие в сфере компьютерной графики произошло существенное развитие в области алгоритмов сжатия изображений. Эта сфера фокусируется на создании новых и улучшении уже существующих методов

сжатия. Основным объектом данной области являются изображения, которые отличаются тремя ключевыми характеристиками.

1. «Изображения занимают намного больше места в памяти, чем текст. Так, скромная, не очень качественная иллюстрация на обложке книги размером 500 × 800 точек, занимает 1.2 Мб – столько же, сколько художественная книга из 400 страниц (60 знаков в строке, 42 строки на странице). Эта особенность изображений определяет актуальность алгоритмов архивации графики» [2, с. 272].

2. Контурные линии и изменения цветовых оттенков на изображении привлекают основное внимание человеческого зрения, в то время как незначительные изменения не вызывают столь яркой реакции. Именно это свойство позволяет разработать эффективные алгоритмы сжатия изображений, которые обеспечивают минимальные потери качества при декомпрессии.

3. «В отличие от текста, изображения обладают избыточностью в двух измерениях, так как соседние точки в изображении имеют близкие цвета как по горизонтали, так и по вертикали. Кроме того, существуют сходства между цветовыми каналами R, G и B, которые могут быть использованы для разработки более эффективных алгоритмов сжатия изображений без потери качества» [2, с. 272].

Существует два типа сжатия изображений: сжатие с потерей качества и без потери качества.

Когда речь идет об искусственно созданных изображениях, таких как графика и иконки программ, а также изображениях, которые будут обработаны алгоритмами распознавания, наиболее предпочтительным является сжатие без потерь. Использование алгоритмов сжатия с потерями может приводить к появлению артефактов, которые могут быть заметны человеческим глазом при повышении степени сжатия.

1.2.1 Сжатие изображений без потерь

Сжатие без потерь заключается в удалении ненужных битов из

исходного файла. В результате получается файл меньшего размера, но с тем же качеством изображения, что и в оригинале.

Существует пять алгоритмов сжатия изображений без потери.

1. «Алгоритм RLE (Run Length Encoding) является одним из самых простых методов сжатия графических данных. Он преобразует изображение в последовательность байтов, расположенных по строкам растра, и затем сжимает данные, заменяя повторяющиеся последовательности байтов на пары значений (количество повторов, значение)» [2, с. 289]:

- сначала инициализируется процесс декомпрессии;
- затем цикл повторяется, пока не будет достигнут конец файла;
- в каждой итерации цикла, читается следующий байт из входного файла;
- если байт является счетчиком, то из входного файла читается следующий байт, который используется в качестве значения, и записывается в выходной файл несколько раз;
- если байт не является счетчиком, то он записывается в выходной файл. Для определения счетчика повторений в RLE-алгоритме используется признак, который устанавливается в двух верхних битах и принимает значение единицы.

Для отображения счетчика, который может принимать значения от 1 до 64, используются оставшиеся 6 бит байта. Это позволяет сжать 64 повторяющихся байта до всего двух, что приводит к значительному уменьшению данных на 32 раза.

«Решение проблемы достигается обычно простановкой меток вначале кодированных цепочек. Такими метками могут быть, например, характерные значения битов в первом байте кодированной серии, значения первого байта кодированной серии и т.п. Лучший, средний и худший коэффициенты сжатия равны 1/32, 1/2 и 2/1» [19, с. 15].

Данный метод сжатия данных может быть использован для сжатия

большинства растровых форматов, таких как TIFF, BMP и PCX Он был специально разработан для деловой графики, которая включает в себя изображения с большим количеством повторяющихся цветов:

- вторая реализация данного алгоритма обеспечивает более высокий максимальный коэффициента сжатия, при этом не увеличивая размер исходного файла так сильно, как первая;
- алгоритм инициализирует необходимые переменные и открывает файл изображения для чтения и сжатый файл для записи;
- затем алгоритм входит в цикл, который читает каждый байт из файла изображения, пока не достигнут конец файла;
- для каждого прочитанного байта алгоритм проверяет, является ли он повторяющимся байтом или нет;
- если байт является повторяющимся, алгоритм считывает следующий байт и записывает его в сжатый файл несколько раз, соответствующее значению счётчика;
- если байт не является повторяющимся, алгоритм считывает следующие counter байт из файла изображения и записывает их в сжатый файл;
- наконец, алгоритм записывает в сжатый файл байт, который был считан в начале цикла;
- цикл продолжается, пока не будет достигнут конец файла изображения.

Данный алгоритм использует старший бит каждого байта для определения возможности повторения данных. В наилучшем случае он способен уменьшить размер файла в 64 раза, а в наихудшем – увеличить его только на 1/128. Средние показатели соответствуют первому варианту.

Преимуществом данного алгоритма является его способность быстро обрабатывать данные без необходимости дополнительного использования памяти. Кроме того, групповое кодирование изображений в формате PCX

позволяет путем изменения порядка цветов в палитре изображения, увеличить степень сжатия без потери качества.

Метод RLE имеет недостатки, поскольку он не всегда эффективен при сжатии данных с небольшим количеством серий и повторяющихся байтов, что может привести к высокой стоимости кодирования.

2. Алгоритм LZW использует сжатие посредством одинаковых цепочек байтов. Процесс состоит в последовательном считывании символов и поиске соответствующих строк в таблице. Если строка отсутствует в таблице, то она добавляется в нее, а затем код для последней найденной строки выводится в выходной поток.

Для заполнения таблицы используется метод InitTable(), который очищает ее и заполняет всевозможными строками, содержащими один символ. Таблица получает начальные значения, в которых присутствуют все возможные комбинации строк, состоящих из одного символа.

«Функция ReadNextByte() считывает символ из файла, а функция WriteCode() записывает соответствующий код в выходной файл. Функция AddStringToTable() добавляет новую строку с присвоением ей уникального кода. Если таблица заполнена до конца, то происходит вывод кода для предыдущей строки и кода очистки, после чего таблица инициализируется функцией InitTable(). Функция CodeForString() возвращает код для найденной строки в таблице» [2, с. 293].

Алгоритм обладает коэффициентами сжатия, равными 1/1000, 1/4 и 7/5. Оптимальное сжатие достигается на изображениях, объем которых превышает 4 Мб. В некоторых редких случаях алгоритм может привести к увеличению размера изображения. Алгоритм LZW симметричен и универсален, применяется в архиваторах.

«Отличительной чертой метода LZW является то, что для распаковки данных нет необходимости хранить таблицу строк в файле. Алгоритм построен так, что таблицу строк можно восстановить с помощью потока кодов, добавляя для каждого кода строку, состоящую из уже существующей в

таблице строки и символа, с которого начинается следующая строка в потоке» [2, с. 294]:

$$C_n, C_{n+1}, C_{n+2}, C_{n+3}, C_{n+4}, C_{n+5}, C_{n+6}, C_{n+7}, C_{n+8}, C_{n+9}, \quad (1)$$

где $C_n, C_{n+1}, C_{n+2}, C_{n+3}, C_{n+4}$ – код этой строки добавляется в таблицу;

$C_n, C_{n+1}, C_{n+2}, C_{n+3}, C_{n+4}, C_{n+5}, C_{n+6}, C_{n+7}, C_{n+8}, C_{n+9}$ – коды этих строк идут в выходной поток.

Данный алгоритм отличается высокой скоростью работы при упаковке и распаковке, небольшими требованиями к памяти и обычной аппаратной реализацией. Реализован в форматах TGA, GIF и TIFF.

Недостатком метода LZW является невысокая степень сжатия ниже, чем при двухэтапном кодировании. В результате работы этого метода поток входящих символов преобразуется на выходе в поток индексов ячеек словаря.

3. Алгоритм Хаффмана – это алгоритм, который не используется для сжатия изображений в чистом виде, но является базовой стадией в более сложных схемах. Модифицированный вариант этого алгоритма применяется для сжатия черно-белых изображений (1 бит на пиксель). Для этого последовательность черных и белых точек заменяется числом, а последовательность сжимается с помощью алгоритма Хаффмана с использованием фиксированной таблицы:

$$A = \{a_1, a_2, \dots, a_n\}, \quad (2)$$

где A – алфавит из n различных символов.

$$W = \{w_1, w_2, \dots, w_n\}, \quad (3)$$

где W – соответствующий ему набор положительных целых весов.

Тогда набор бинарных кодов выглядит следующим образом:

$$C = \{c_1, c_2, \dots, c_n\}, \quad (4)$$

где c_i – является кодом для символа a_i;

c_i – не является префиксом для c_j, при $i \neq j$.

«Алгоритм Хаффмана при сжатии изображений используется для учета частоты появления одинаковых байтов в изображении. Это позволяет сопоставить пикселям изображения коды разной длины в зависимости от их частоты встречаемости. Пикселям, встречающимся чаще, сопоставляются коды наименьшей длины, а реже встречающимся – коды большей длины. Для сбора статистики требуется выполнить две итерации по файлу: одну – для сбора статистической информации, вторую – для кодирования. Коэффициенты сжатия: 1/8, 2/3, 1. Применяя такой метод, потребуется запись в файл таблицы соответствия закодированных пикселей и кодирующих последовательностей. Такое кодирование используется в качестве заключительного этапа архивации в формате JPEG и обеспечивает довольно высокую скорость и хорошее качество» [2, с. 34].

Одним из недостатков алгоритма Хаффмана является его зависимость от близости вероятностей символов к степени двойки, что может привести к невозможности сжатия данных, если вероятности близки к 1/2. Кроме того, для кодирования каждого символа необходимо использовать целое число бит, что приводит к неэффективному использованию памяти. Алгоритм Хаффмана для сжатия и передачи изображений использует формат TIFF.

4. «Алгоритм JBIG – алгоритм был специально разработан ISO (Joint Bi-level Experts Group) для сжатия черно-белых изображений, включая факсовые и отсканированные документы, используя только 1 бит на пиксель. Кроме того, он может применяться для 2-х и 4-х битовых изображений, разбивая их на отдельные битовые плоскости. Одним из преимуществ JBIG является возможность управления параметрами изображения такими как порядок разбиения изображения на битовые плоскости, ширина полос в

изображении и уровни масштабирования. Это делает его очень эффективным для работы с большими объемами изображений» [2, с. 302].

Параметры алгоритма могут быть настроены таким образом, что создается эффект «огрубленного изображения», полезный при передаче изображений по сети с низкой пропускной способностью. В этом случае изображение медленно появляется на экране, что позволяет человеку изучать картинку до окончания процесса распаковки. Одним из преимуществ JBIG является возможность создания уменьшенных копий изображений, что обеспечивает быстрый доступ к базе данных изображений.

Отличие данного алгоритма от алгоритма Хаффмана заключается в использовании последовательностей символов. Алгоритм создает короткие цепочки для часто встречающихся символов и длинные цепочки для редко встречающихся символов.

Одной из особенностей JBIG является снижение коэффициента сжатия при наличии шумов в исходном изображении.

5. Алгоритм Lossless GPEG – был создан командой в области фотографии (Joint Photographic Expert Group). Данный алгоритм обрабатывает полноцветные изображения в градациях серого 8 или 24 битов без использования цветовой палитры. Он имеет три уровня сжатия без потерь: 20, 2 и 1.

«Стандарт сжатия изображений JPEG поддерживает два варианта сжатия: с потерями и без потерь. Для сжатия без потерь используется метод разностного кодирования, основанный на высокой корреляции между точками изображения. Это позволяет сжимать последовательности точек изображения x_1, x_2 ,...., x_N, путем кодирования последовательности их разностей» [2, с. 303]:

$$y_i = x_i - x_{i-1}, \qquad i = 1, 2, \dots N, \qquad x_0 = 0, \tag{5}$$

где y_i– числа с ошибками;

x_i – предсказания.

«Стандарт Lossless JPEG учитывает формирование ошибок предсказания с использованием предыдущих закодированных точек в текущей или предыдущей строке. Это позволяет более эффективно сжимать изображения без потери качества, чем другие алгоритмы сжатия» [2, с. 303].

Lossless JPEG рекомендуется использовать в тех приложениях, где необходимо сохранить битовую точность между исходным и разархивированным изображением.

Сжатие без потерь может уменьшить размер изображения примерно в два раза, в то время как сжатие с потерями может сократить размер изображения от 10 до 200 раз.

«Одной из причин большой разницы между сжатием без потерь и с потерями является то, что классические алгоритмы работают с цепочками и не учитывают так называемую «когерентность областей» в изображениях. Эта концепция основана на незначительных изменениях цвета и структуры изображения в небольшой области» [2, с. 303].

Алгоритмы сжатия без потерь могут получить более высокое качество изображения без потерь при минимальном размере файла.

1.2.2 Сжатие изображений с потерями

Сжатие с потерями – это метод сжатия данных, при котором распакованные данные отличаются от исходных, однако эти отличия обычно несущественны для большинства приложений. Существует четыре алгоритма сжатия изображений с потерями:

1. Алгоритм GPEG – это мощный алгоритм для сжатия полноцветных изображений, который стал де-факто стандартом. Он использует косинусное преобразование для разложения изображения на амплитуды частот. Это приводит к тому, что большинство коэффициентов матрицы изображения становятся близкими к нулю или равными ему. Также применяется квантование коэффициентов, что позволяет достигать больших коэффициентов сжатия. В результате сжатия сохраняется плавность

изменения цветов в изображении, а потери качества остаются незначительными благодаря несовершенству человеческого зрения. Этот алгоритм был создан специально для сжатия изображений, содержащих 24 бита, и является стандартом Joint Photographic Expert Group (JPEG) ISO.

Работа алгоритма:

1. Для преобразования изображений из цветового пространства *RGB*, которое использует компоненты Red, Green и Blue, в цветовое пространство *YUV*, требуется выполнить соответствующую операцию. Преобразование форматов из *RGB* в *YUV* используется формула:

$$\begin{vmatrix} Y \\ U \\ V \end{vmatrix} = \begin{vmatrix} 0.299 & 0.587 & 0.114 \\ 0.5 & -0.4187 & -0.0813 \\ 0.1687 & -0.3313 & 0.5 \end{vmatrix} \cdot \begin{vmatrix} R \\ G \\ B \end{vmatrix} + \begin{vmatrix} 0 \\ 128 \\ 128 \end{vmatrix}, \quad (6)$$

а для преобразования форматов из *YUV* в *RGB* используется формула:

$$\begin{vmatrix} R \\ G \\ B \end{vmatrix} = \begin{vmatrix} 1 & 0 & 1.402 \\ 1 & -0.34414 & -0.71414 \\ 1 & 1.772 & 0 \end{vmatrix} \cdot \left(\begin{vmatrix} Y \\ U \\ V \end{vmatrix} - \begin{vmatrix} 0 \\ 128 \\ 128 \end{vmatrix} \right), \quad (7)$$

где Y – яркостная составляющая;

U, V – компоненты, которые отвечают за цвет.

Возможно использование методов сжатия с большими потерями для массивов U и V, что приводит к достижению более высоких коэффициентов сжатия, из-за того, что человеческий глаз менее чувствителен к цветовой информации, чем к информации о яркости.

2. При больших коэффициентах сжатия процесс становится сложнее. Для сжатия изображения мы применяем ДКП к матрицам 8×8, получая три рабочие матрицы по 8 бит для каждой компоненты цвета. Компонента Y разбивается на матрицы, а компоненты U и V формируются из исходной матрицы размером 16×16. Таким образом, изображение сжимается в два раза

благодаря использованию пространства цветов *YUV*, однако теряется 3/4 полезной информации о цветах. В результате получается одна рабочая матрица ДКП, содержащая информацию о цветовых составляющих изображения. В итоге качество *RGB* изображения не сильно страдает.

3. Применяя ДКП к каждой из трех рабочих матриц, мы получаем матрицы коэффициентов, в которых находятся низкочастотные составляющие в левом верхнем углу, а высокочастотные – в правом нижнем.

Преобразование в упрощенном виде выглядит следующим образом:

$$Y(u,v) = \frac{1}{4} \cdot \sum_{i=0}^{7} \sum_{j=0}^{7} C(u) \cdot C(v) \cdot A(u) \cdot cos\left(\frac{(2i+2) \cdot u\pi}{2n}\right), \quad (8)$$

$$A(u) = \begin{cases} \frac{1}{\sqrt{2}}, for\ u \equiv 0 \\ 1, for\ u \neq 0 \end{cases}, \quad (9)$$

где $Y(u,v)$ – коэффициент DCT для блока 8x8 пикселей;

u, v – индексы частоты в диапазоне 0...7;

$A(u)$ – значение яркости пикселя.

4. Квантование. Для каждой компоненты (*Y*, *U* и *V*) необходимо определить свою матрицу квантования $q[u,v]$, поэлементно разделить рабочую матрицу на неё и выполнить квантование матрицы:

$$Yq(u,v) = IntegerRound \cdot \left(\frac{Y[u,v]}{q[u,v]}\right). \quad (10)$$

«Наибольшие потери при сжатии изображений в формате JPEG происходят на этапе использования матриц коэффициентов и управления степенью сжатия. Стандарт JPEG включает рекомендованные матрицы, полученные опытным путем, а матрицы для разных коэффициентов сжатия получаются путем умножения исходной матрицы на число gamma. При

использовании больших значений gamma могут возникнуть эффекты алгоритма, такие как разделение изображения на квадраты 8×8 или эффект Гиббса в высоких частотах, проявляющийся в виде нимба вокруг контуров с резким переходом цвета» [2, с. 308].

5. Для преобразования матрицы 8×8 в 64-элементный вектор можно использовать метод «зигзаг» – способ сканирования элементов, при котором они выбираются по индексам (0,0), (0,1), (1,0), (2,0) и так далее.

После применения метода «зигзаг» для сканирования матрицы, полученный вектор обладает низкочастотными коэффициентами в начале и высокочастотными в конце.

6. «Для свертки вектора можно воспользоваться алгоритмом группового кодирования, который преобразует вектор в пары чисел вида <пропустить, число>. Значение «пропустить» показывает количество нулевых элементов, а «число» – значение следующего ненулевого элемента» [2, с. 309].

7. Кодирование по Хаффману с использованием фиксированной таблицы используется для дальнейшего сжатия пар, полученных на предыдущем этапе. Этот метод сжатия позволяет сократить объем изображения на 10-15 раз без значительных потерь в качестве, а процесс восстановления изображения является полностью симметричным.

2. «Фрактальный алгоритм – основан на том, что мы представляем изображение в более компактной форме – с помощью коэффициентов системы итерируемых функций (Iterated Function System – IFS)» [2, с. 311].

IFS используется для преобразования одного изображения в другое. Эти преобразования применяются к точкам, находящимся в трехмерном пространстве, которые определяются координатами *x* и *y*, а также яркостью.

Экран с изображением и система линз, которые могут проецировать выбранные части изображения в различные места на другой экран, включены в концепцию фотокопировальной машины, описанной в книге «Fractal Image Compression» Барнсли. Кроме того, эти линзы могут изменять яркость и

контрастность изображения, зеркально отображать и поворачивать его фрагменты, а также масштабировать их.

Путем изменения характеристик линз мы можем контролировать получаемое изображение. Итерация фотокопировальной машины строится на основе предыдущего изображения, что приводит к возникновению фрактала, зависящего от параметров и позиций линз.

«Фрактальная компрессия – это поиск самоподобных областей в изображении и определение для них параметров аффинных преобразований» [2, с. 313].

«Преобразование $f: X \to X$, в метрическом пространстве (X, d) называется сжимающим, если существует число s: $0 \leq s < 1$, такое, что» [2, с. 314]

$$d\big(f(x), f(y)\big) \leq sd(x, y). \tag{11}$$

Построение алгоритма:

- «для создания многообразия геометрических фигур, используется квадраты со сторонами, параллельными сторонам изображения;
- разбиваем изображение на квадратные доменные блоки одинакового размера. Для упрощения процесса сжатия и распаковки мы уменьшаем размер блоков в два раза при переводе из доменной области в ранговую;
- получаем доменные области «через точку» по осям X и Y. Этот подход позволяет сократить перебор в 4 раза и ускорить процесс обработки изображений;
- для преобразования доменной области в ранговую мы используем поворот куба на 0°, 90°, 180° или 270° или зеркальное отражение. Общее количество возможных преобразований составляет 8 штук;
- масштабируем изображение по вертикали (яркости) в

фиксированное количество раз, а именно в 0,75» [2, с. 316].

3. Рекурсивный (wavelet) алгоритм – этот вид сжатия использует когерентные области для цветных и черно-белых изображений с плавными переходами. Коэффициент сжатия может варьироваться от 5 до 100, при этом при больших значениях могут появляться нежелательные «лестничные эффекты» на резких границах, особенно по диагонали.

Рекурсивный алгоритм сжатия изображений отличается от других методов тем, что он использует более сложный подход к обработке блоков. Алгоритм сохраняет в файл разницу между средними значениями соседних блоков, которая почти всегда близка к 0.

Так два числа a_{2i} и a_{2i+1} могут быть представлены в виде:

$$b_i^1 = \frac{a_{2i} + a_{2i+1}}{2}, \tag{12}$$

$$b_i^2 = \frac{a_{2i} - a_{2i+1}}{2}. \tag{13}$$

Последовательность a_i может быть преобразована в последовательность $b_i^{1,2}$, которая будет состоять из парных элементов.

Для обработки двумерных данных алгоритм реализуется аналогично. Если у нас имеется квадрат из четырех точек с яркостями $a_{2i,2j}, a_{2i+1,2j}, a_{2i,2j+1}, a_{2i+1,2j+1}$, то

$$\begin{aligned} b_{ij}^1 &= \frac{a_{2i,2j} + a_{2i+1,2j} + a_{2i,2j+1} + a_{2i+1,2j+1}}{4}, \\ b_{ij}^2 &= \frac{a_{2i,2j} + a_{2i+1,2j} - a_{2i,2j+1} - a_{2i+1,2j+1}}{4}, \\ b_{ij}^3 &= \frac{a_{2i,2j} - a_{2i+1,2j} + a_{2i,2j+1} - a_{2i+1,2j+1}}{4}, \\ b_{ij}^4 &= \frac{a_{2i,2j} - a_{2i+1,2j} - a_{2i,2j+1} + a_{2i+1,2j+1}}{4}. \end{aligned} \tag{14}$$

Одно из преимуществ данного алгоритма заключается в том, что он позволяет постепенно передавать изображение по сети. В начале изображение передается в сжатом виде, что упрощает показ «грубой» версии изображения.

В отличие от JPEG и фрактального сжатия, данный алгоритм использует блоки различных размеров, таких как 2×2, 4×4, 8×8 и т.д. Коэффициенты для каждого блока сохраняются независимо, что позволяет избежать «мозаичного» эффекта на изображении, который может присутствовать в результате использования других методов сжатия

В заключении параметры алгоритмов сжатия изображений будут представлены в таблице 1.

Таблица 1 – Параметры алгоритмов

Алгоритм	Особенности сжатия	Коэффициенты сжатия	Симметричность времени	На что ориентирован	Потери	Размерность
RLE	Подряд идущие одинаковые цвета: 2 2 2 2 2 2 15 15 15	32, 2, 0.5	1	3,4-х битные	Нет	1D
LZW	Одинаковые подцепочки: 2 3 15 40 2 3 15 40	1000, 4, 5/7	1.2-3	1-8 битные	Нет	1D
Хаффмана	Разная частота появления цвета: 2 2 3 2 2 4 3 2 2 2 4	8, 1.5, 1	1-1.5	8 битные	Нет	1D
JBIG	Преобладание белого цвета в изображении, большие области, заполненные одним цветом	2-30 раз	~1	1 битные	Нет	2D
Lossless GPEG	Отсутствие резких границ	2 раза	~1	24 битные серые	Нет	2D
GPEG		2-20 раз	~1	24 битные серые	Да	2D
Рекурсивный	Плавные переходы цветов и отсутствие резких границ	2-200 раз	1.5	24 битные серые	Да	2D
Фрактальный	Подобие между элементами изображения	2-2000 раз	1000-10000	24 битные серые	Да	2.5D

В таблице алгоритмов сжатия изображения были рассмотрены:

- ориентация н на создание фотореалистичных изображений с использованием 16 миллионов цветов (24 бита);
- использование сжатия с потерями, что дает возможность регулировать качество сжатых изображений;
- использование избыточности изображений в двух плоскостях;
- появление существенно несимметричных алгоритмов;
- увеличивающаяся степень сжатия изображений.

Выводы по первому разделу

В первом разделе происходит исследование предметной области. Исследуем и анализируем существующие алгоритмы сжатия изображений с потерями качества и без потери качества. Выявляем преимущества данных алгоритмов и их недостатки. В заключении, представлен сравнительный анализ алгоритмов сжатия изображения.

2 Проектирование мобильного приложения

Результатом данной работы должно являться создание алгоритма сжатия для оперативной съемки и отправки проблемы для АХО ТГУ.

Чтобы выполнить поставленную задачу требуется:

- определить основные инструментальные средства для разработки мобильного приложения;
- сравнить и выбрать алгоритм сжатия изображения для реализации модуля оперативной съемки;
- спроектировать мобильное приложение;
- спроектировать модуль оперативной съемки.

2.1 Структура мобильного приложения

Android-приложения может иметь разную сложность, но структура всегда имеет постоянный набор элементов, как обязательных, так и необязательных, которые можно использовать по необходимости.

Основными компонентами Android-приложения являются манифест приложения, набор ресурсов и исходный код программы:

- src – каталог, который содержит исходный код приложения;
- com.example.StructureProjectApp – это package, название которого, указывается при создании проекта;
- Main.java – файл исходного кода, в котором описан класс главной Activity;
- gen – содержит java файлы, которые создаются автоматически во время разработки приложения. Здесь расположен файл «R.java»;
- BuildConfig.java – содержит константу «DEBUG», значение которой устанавливается автоматически в зависимости от того, является ли версия приложения отладочной или релизной;
- R.java – это файл, который содержит уникальные идентификаторы для всех ресурсов приложения (ID). Он используется для доступа к

ресурсам из кода приложения;

- main.xml – файл, который описывает сборку элементов экрана. Он используется для создания пользовательского интерфейса приложения;
- assets – папка, позволяющая организовать произвольную структуру вложенных каталогов и файлов. Она используется для хранения данных, которые необходимы приложению во время выполнения;
- bin – каталог сборки приложения;
- res – каталог, содержащий структуру папок ресурсов приложения. Наименование каждой папки определяет тип ресурсов, которые она должна содержать. В данном каталоге могут присутствовать подпапки с названиями drawable, menu, anim, layout, xml, values и raw. Они используются для хранения различных типов ресурсов, таких как изображения, меню, анимации, макеты и т.д;
- AndroidManifest – файл, который показывает точку входа в приложение;
- protect.properties – содержит параметры проекта, например, версию. Он используется для настройки проекта и его сборки.

Android включает в себя четыре основных типа компонентов: Activities, Broadcast receivers, Services и Content providers.

«Activity – это компонент в Android представляет пользовательский интерфейс для определенного действия, которое может быть выполнено пользователем. Каждый экран приложения содержит свой Activity, который не зависит от других Activity. Activity может находиться в одном из трех состояний» [3, с. 3]:

- «active или running – находится на переднем плане и имеет фокус для взаимодействия с пользователем;
- paused – не имеет фокуса, но отображается пользователю. Может быть перекрыто другим Activity. Сохраняет свое состояние, но может

быть уничтожено системой при нехватке памяти. В состоянии stopped не отображается и также может быть уничтожено системой для освобождения памяти;

– stopped – полностью перекрыто другим Activity и больше не отображается пользователю. Это состояние может быть вызвано либо системой, либо другими Activity. В случае нехватки памяти Activity в состоянии stopped может быть уничтожено системой» [3, с. 3].

Activity уведомляет систему о переходе между состояниями, вызывая соответствующие методы. Если Activity приостанавливается или останавливается, система может удалить его из памяти, запросить завершение или уничтожить процесс. При следующем отображении Activity состояние будет восстановлено полностью.

Существует жизненный цикл приложения. После запуска Activity проходит через ряд событий, которые обрабатываются системой и для обработки которых существует ряд обратных вызовов:

– void onCreate() – первый метод, который вызывается при запуске Activity и переводит его в состояние Created. Здесь происходит первоначальная настройка Activity, в том числе создание объектов визуального интерфейса. Путем вызова данного метода можно получить объект Bundle, который содержит сохраненное состояние Activity. В случае, если Activity создается впервые, объект Bundle содержит значение null. Если же Activity ранее уже была создана и находилась в приостановленном состоянии, то объект Bundle содержит связанную с ней информацию. После завершения метода onCreate(), Activity переходит в состояние Started, система автоматически вызывает метод onStart();

– void onStart() – выполняет подготовку к отображению Activity на экране устройства. Встроенный код этого метода обычно не требует

переопределения, поскольку он выполняет всю необходимую работу. После выполнения метода, Activity отображается на экране устройства, переходит в состояние Resumed и вызывается метод onResume();

- void onResume() – переводит Activity в состояние Resumed и отображает его на экране устройства, что позволяет пользователю взаимодействовать с ним. Activity находится в состоянии Resumed до тех пор, пока не потеряет фокус, например, из-за переключения на другую Activity или выключения экрана устройства;
- void onPause() – при переходе пользователя на другую Activity система вызывает метод onPause(), а текущая Activity переходит в состояние Paused. В этом методе можно освободить используемые ресурсы, временно остановить процессы, такие как воспроизведение аудио или анимации, приостановить работу камеры и т.д., чтобы они не негативно сказывались на производительности системы. Однако, в данном случае Activity все еще видима на экране, поэтому метод onPause() выполняется быстро. Когда метод onPause() завершается, текущая Activity остается активной, но становится невидимой на экране устройства. Если пользователь захочет вернуться к ней, система вызовет метод onResume() и Activity снова будет отображена на экране. Если система обнаружит, что для работы активных приложений нужно больше памяти, она может завершить работу неактивной Activity, которая находится в фоне;
- void onStop() – в данном методе необходимо освобождать ресурсы, которые не используются пользователем при отсутствии взаимодействия с Activity, а также сохранять данные. В то же время, в состоянии Stopped Activity остается в памяти устройства, и сохраняется состояние всех элементов пользовательского интерфейса. Если пользователь решит вернуться к предыдущей

Activity, система вызовет метод onRestart(). Если Activity завершает свою работу, например, при закрытии приложения, то система вызывает метод onDestroy();

- void onDestroy() – завершается работа activity вызовом метода onDestroy, который появляется, или в случае если система решит уничтожить activity, если возникли конфигурационные причини (поворот экрана или при многооконном режиме), или при вызове метода finish(). Также следует обозначить, что при изменении ориентации экрана система завершает activity и вслед затем создает ее заново, вызывая метод onCreate;
- void onRestart() – этот метод вызывается перед методом onStart(), если Activity восстанавливается из состояния Stopped, а не создается заново. Этот метод позволяет выполнить необходимые действия для подготовки Activity к его повторному запуску после перехода из состояния Stopped.

На рисунке 2 представлена блок-схема алгоритма жизненного цикла Activity.

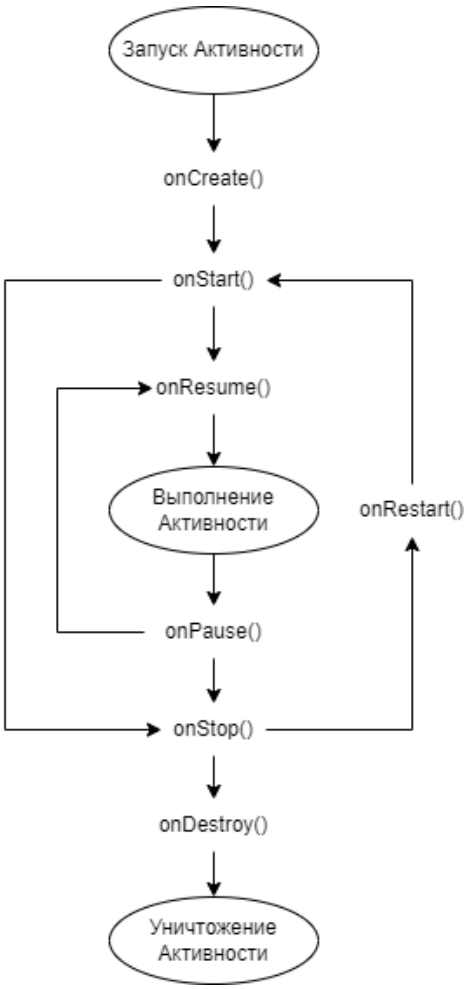

Рисунок 2 – Жизненный цикл Activity

Service – представляет собой фоновый процесс, который может выполнять длительные вычисления и получать данные по сети. Один из примеров его использования – проигрывание музыки в фоновом режиме, даже если связанная с ней Activity закрыта. Как и у Activity, у Service есть свой жизненный цикл и соответствующие методы:

- void onCreate();
- void onStart(Intent intent);
- void onDestroy().

Аналогично Activity, запуск Service происходит в главном потоке процесса приложения.

«Broadcast receiver – представляет собой компонент приложения, который не выполняет никаких действий, кроме рассылки и отклика

широковещательных сообщений, например, о низком заряде батареи» [3, с. 3].

Broadcast Receiver не обладает пользовательским интерфейсом, однако может запустить Activity в ответ на полученное сообщение или использовать NotificationManager для привлечения внимания пользователя. Для этого могут быть использованы различные способы, включая вибрацию, звуковые сигналы или мигание вспышки.

«Приемник широковещательных сообщений содержит только один метод жизненного цикла: onReceive(). Когда приемник получает широковещательное сообщение, операционная система вызывает метод onReceive() и передает ему намерение с сообщением. Региональный трансляционный приемник активен только во время выполнения этого метода. Процесс, который в настоящее время выполняет передатчик, считается приоритетным и будет сохранен» [3, с. 3].

После завершения метода onReceive() приемник становится неактивным, и операционная система считает, что объект передатчика завершил свою работу.

Content Provider – это компонент приложения, который обеспечивает доступ к данным, таким как чтение, добавление и обновление. Помимо этого, поставщик контента может предоставлять доступ к данным не только своему приложению, но и другим приложениям. Данные могут храниться как в базе данных, так и в файловой системе.

2.2 Выбор алгоритма сжатия изображения в мобильном приложении

Прежде чем начать реализацию модуля оперативной съемки, необходимо выбрать алгоритм, который будет использоваться в ее основе. После изучения структуры алгоритмов сжатия изображения, а также их преимуществ и недостатков, было принято решение сравнить следующие алгоритмы:

– JPEG;

– фрактальный алгоритм;

– рекурсивный (волновой) алгоритм.

Чтобы определиться с выбором алгоритма для сжатия изображения, возьмем полноценное изображения 320 × 320 × RGB – 307.200 байт, представленное на рисунке 3.

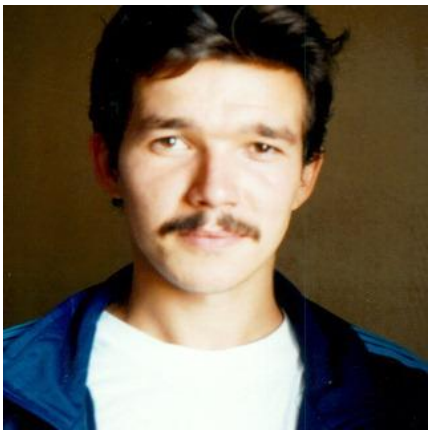

Рисунок 3 – Полноценное изображения 320×320×RGB – 307.200 байт

Для сравнения совершим архивацию изображения в 100 раз JPEG-алгоритмом, фрактальным алгоритмом, рекурсивным (волновым) алгоритмом, представленное на рисунке 4.

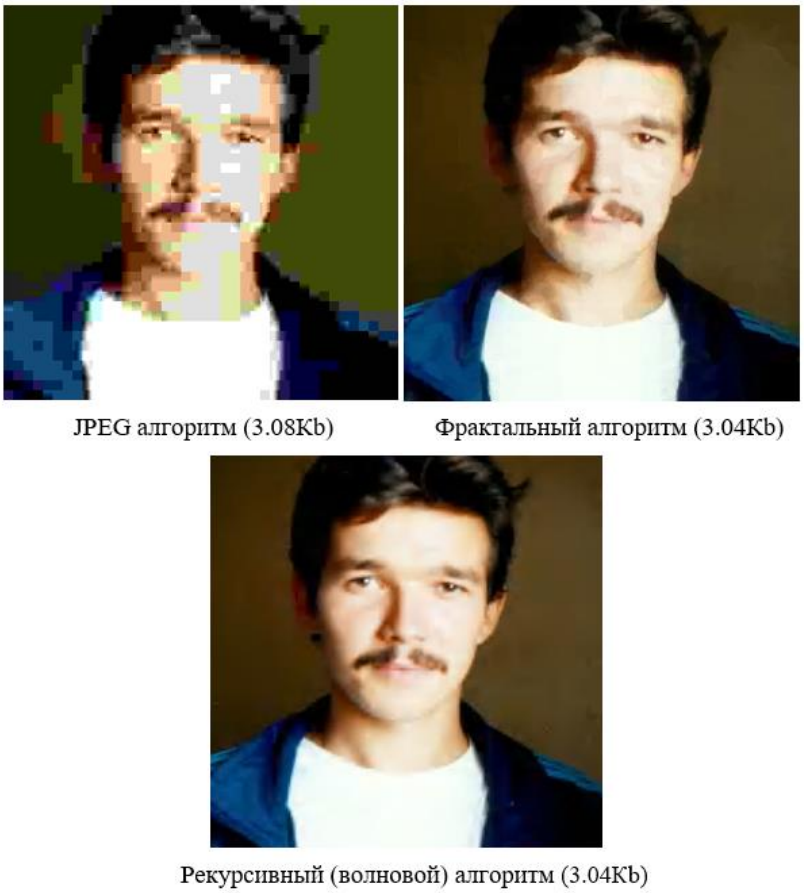

Рисунок 4 – Архивация полноцветного изображения в 100 раз

Из данного примера можно сделать вывод о том, что при использовании высокой степени компрессии, алгоритм JPEG не является конкурентоспособным.

«При визуальном сравнении качество изображения, полученного с помощью фрактального алгоритма, незначительно ниже, чем у других алгоритмов. Однако, для фрактального алгоритма не требуется постобработка изображения, достаточно применить только «разумное» сглаживание, в то время как волновой алгоритм требует более глубокой обработки, что приводит к размытию мелких деталей изображения» [2, с. 369].

Исходя из сказанного выше, был сделан вывод, что у каждого алгоритма есть свои минусы и плюсы, но в разработке модуля оперативной съемки будет взят фрактальный алгоритм, так как обработка с помощью данного алгоритма лучше всего подойдет для распознания деталей на фото с проблемой.

2.3 Проектирование алгоритма мобильного приложения

В начале реализации проектирования мобильного приложения требуется смоделировать диаграмму классов.

Диаграмма классов – это UML-диаграмма, которая описывает систему, отображая различные типы объектов внутри системы и статические связи между ними.

Для отображения структуры приложения в статическом виде используется диаграмма классов, которая также отображает атрибуты и операции, связанные с каждым классом. Основными элементами являются классы и связи между ними.

Классы характеризуются при помощи атрибутов и операций:

- атрибуты определяют свойства объектов класса и должны иметь уникальное имя в рамках класса. За именем атрибута может следовать его тип и значение по умолчанию;
- операция – это функция или процедура, которая может принимать параметры и возвращать значения.

Существуют связи**:**

- ассоциация – это связь между экземплярами классов, которая определяет количество объектов, участвующих в отношении. Ассоциации могут иметь имя и роль, которую выполняют объекты, связанные с концами ассоциации;
- агрегация – это тип ассоциации «целое-часть», который обозначается прямой линией с ромбом на конце. Ромб на конце указывает на агрегирующий класс (т.е. «состоящий из») и агрегированный класс (т.е. «части»);
- «наследование – это отношение типа «общее-частное», которое позволяет определить связь между классами, где один класс имеет поведение и структуру, совпадающие с другими классами. Создание производного класса от базового приводит к появлению иерархии

наследования, а реализация этого принципа позволяет повторно использовать код» [16, с.3].

На рисунке 5 представлена диаграмма классов мобильного приложения АХО ТГУ.

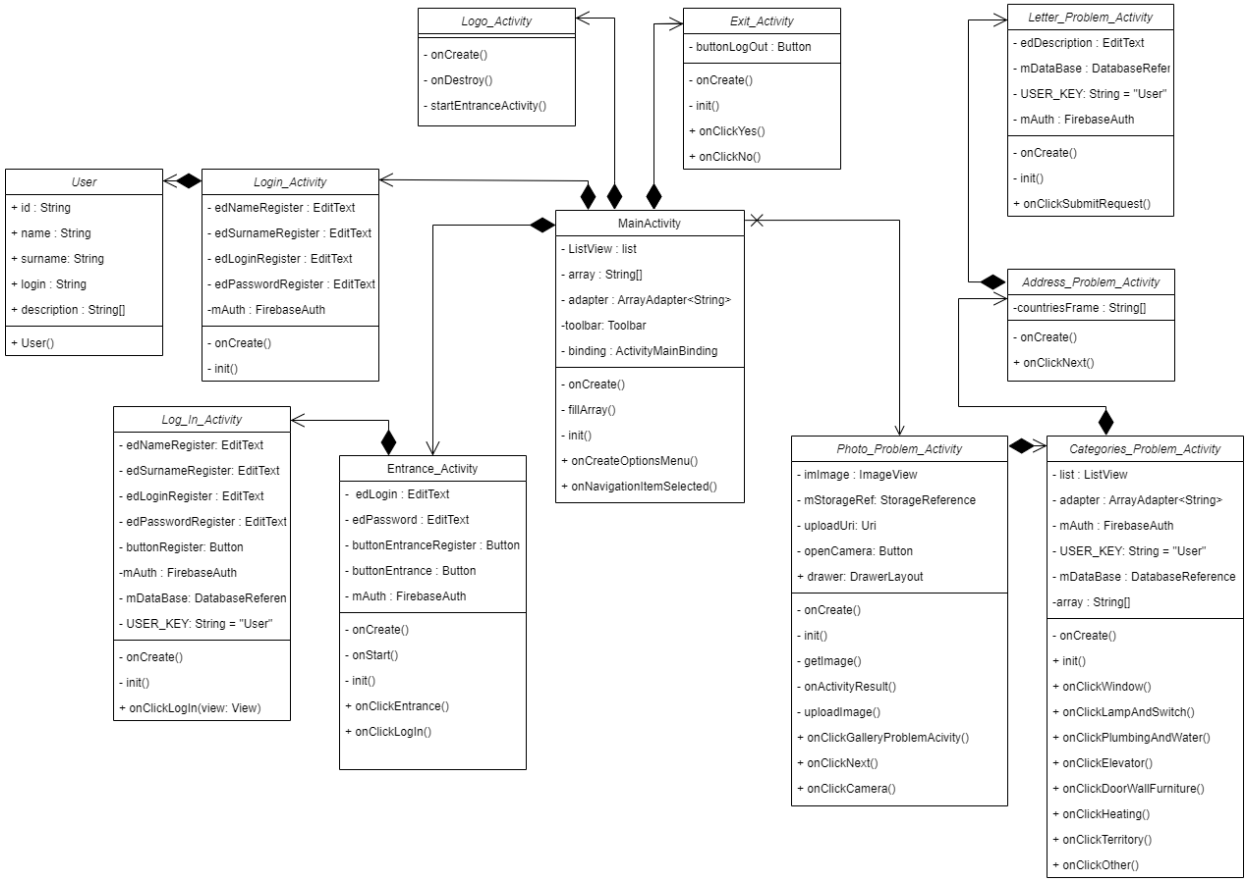

Рисунок 5 – Диаграмма классов разрабатываемого мобильного приложения

На данной диаграмме изображено одиннадцать классов, имеющих между собой отношения наследование. Рассмотрим семантическую составляющую каждого класса:

– Logo_Activity – класс отвечающий за окно с логотипом, которое запускается при входе в данное мобильное приложение;

– Entrance_Activity – класс отвечающий за вход в мобильное приложение. Здесь происходит авторизация пользователя, а также предлагает зарегистрироваться новому пользователю;

– Log_In_Activity – класс отвечающий за регистрацию нового пользователя в мобильном приложении;
– MainActivity – главный класс, в котором реализуется боковое меню из которого можно перейти в основные разделы данного мобильного приложения;
– Photo_Problem_Activity – класс, в котором реализуется метод оперативной съемки в мобильном приложении. В данном классе происходит открытие камеры, с возможностью сделать фото проблемы или загрузка фотографии из галереи пользователя;
– Categories_Problem_Activity – класс, в котором происходит выбор категории и темы категории проблемы;
– Address_Problem_Activity – класс, в котором происходит указание адреса, где произошла проблема;
– Letter_Problem_Activity – класс, в котором указывается дополнительное описание проблемы и происходит подача запроса;
– My_Problems_Activity – класс, в котором отображается все поданные запросы пользователя;
– Exit_Activity – класс, в котором отображается окно выхода.
– User – класс, в котором собираются данные о поданной проблеме пользователя;
– Login_Activity – класс, в котором собираются данные о зарегистрированном пользователе.

Ключевые методы разрабатываемого мобильного приложения:

– onClickLogIn() – функция, в которой происходит процесс регистрации пользователя в мобильном приложении;
– onClickEntrance() – функция, в которой происходит вход пользователя в мобильное приложение;
– onNavigationItemSelected() – функция, в которой реализуется отображение списка разделов в боковом меню;

- onClickCamera() – функция, в которой происходит открытие камеры в мобильном приложении;
- onClickGalleryProblemAcivity() – функция, в которой происходит открытие галереи пользователя в мобильном приложении;
- onActivityResult() – функция, в которой происходит получение URL фотографии пользователя;
- uploadImage() – функция, в которой происходит архивация фотографии;
- onClickNext() – функция перехода на следующие Activity мобильного приложения;
- onClickAll() – функция в которой происходит отображения всех поданных запросов пользователя.

Для проектирования приложений используют блок-схему – это графическое изображение последовательности выполнения операций или алгоритма, в котором блоки соединены линиями и стрелками, показывающими порядок выполнения операций. Она широко используется для визуализации и проектирования процессов, программ и алгоритмов.

Блок-схема мобильного приложения представлена на рисунке 6.

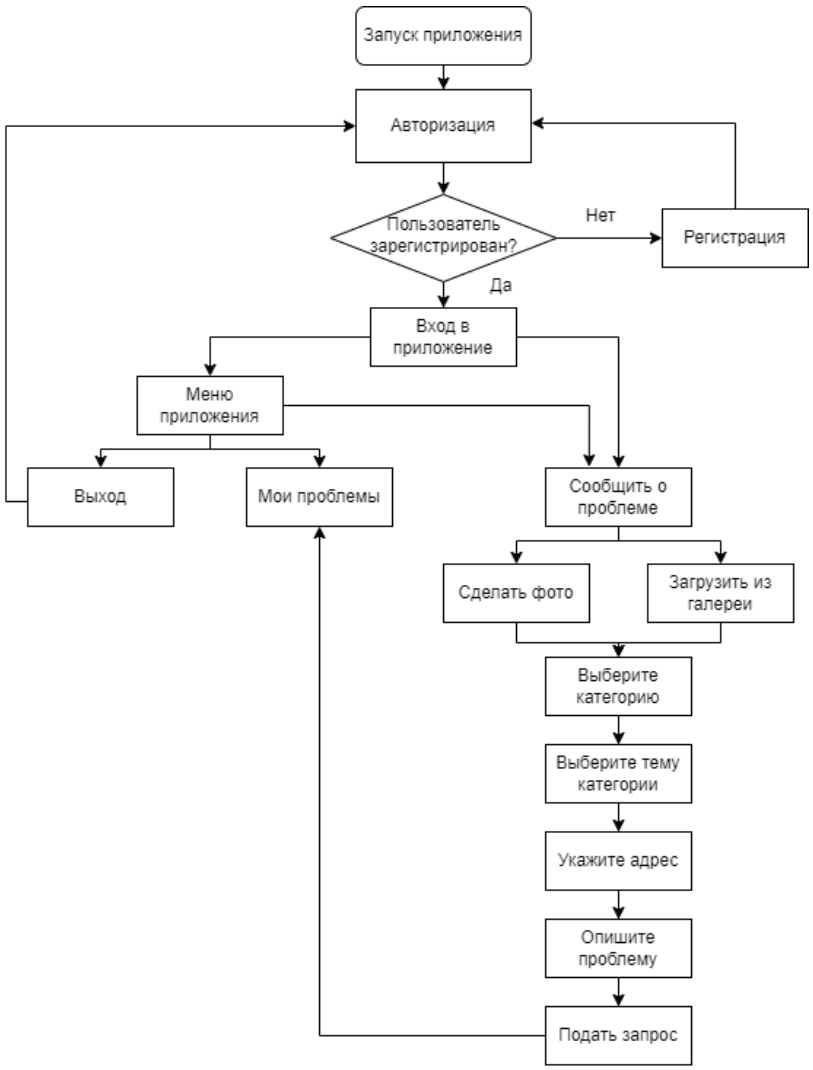

Рисунок 6 – Блок-схема мобильного приложения

На данном рисунке в блок-схеме мобильного приложения показаны процессы, которые происходят при запуске мобильного приложения и вариации его действий.

2.3.3Проектирование модуля оперативной съемки

Мобильное приложение также будет иметь поддерживаемый модуль оперативной съемки, где будет происходить сжатие фотографии с помощью фрактального алгоритма.

Для реализации данного модуля берем за основу фрактальный алгоритм, блок-схема которого представлена на рисунке 7.

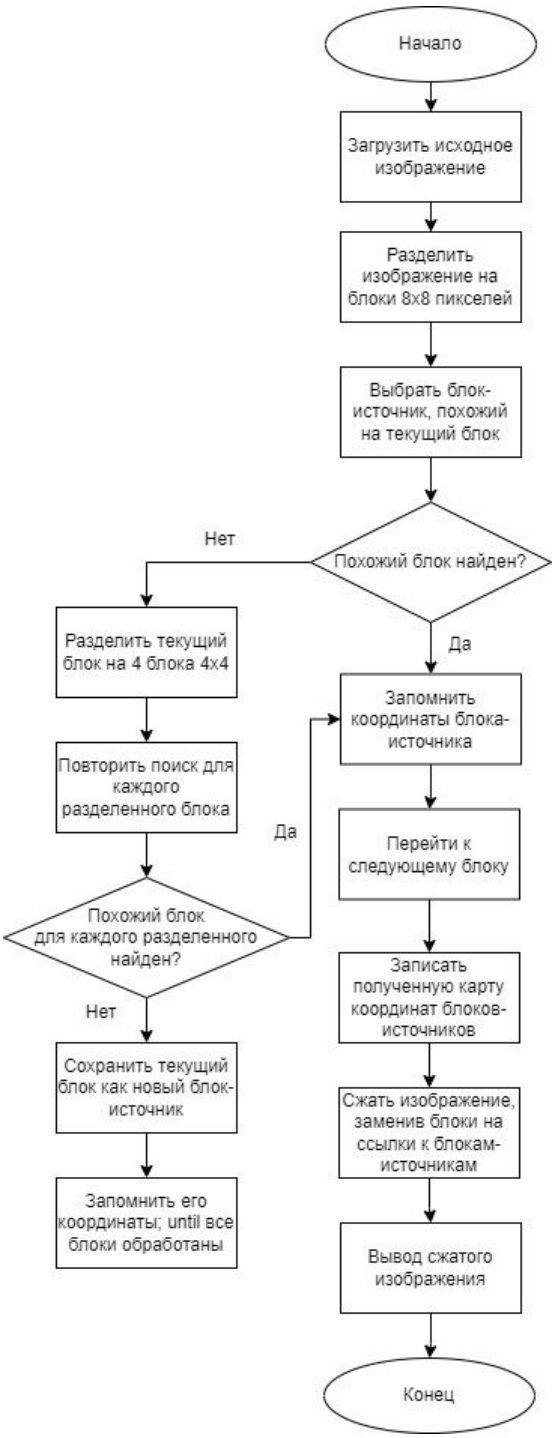

Рисунок 7 – Блок-схема модуля фрактального алгоритма

На данном рисунке в блок-схеме фрактального алгоритма показано, что сначала загружается исходное изображение, которое разбивается на блоки 8×8 пикселей. Затем происходит выбор блок-источника, похожего на текущий блок. Если похожий блок найден – запоминаем его координаты и переходим к

следующему блоку. Если похожий блок не найден – разбиваем текущий блок на 4 блока по 4×4 и повторяем поиск. Если для всех блоков не найдены похожие, то происходит сохранение текущего блока, как нового источника и происходит запоминание его координат. Повторяем данный процесс, пока все блоки не обработаны и записываем карту координат блоков-источников. В конце происходит сжатие изображения, заменив блоки на ссылки к блокам-источникам и выводим сжатое изображение.

Для сжатия изображений с помощью фрактального алгоритма в Java можно использовать библиотеку fractal4j. Изображение загружается из файла «inputImage.jpg». Затем происходит сжатие изображения с помощью фрактального алгоритма, используя метод compressImage() из класса FractalCompressionUtils из библиотеки fractal4j. Готовое сжатое изображение записывается в файл «outputImage.jpg» с помощью метода write() класса ImageIO.

Выводы по второму разделу

В данном разделе происходит выбор основных инструментальных средств для разработки мобильного приложения, сравнение и выбор алгоритма сжатия изображения для реализации модуля оперативной съемки, проектирование мобильного приложения.

Построена модель сжатия изображения с помощью фрактального алгоритма и разработана блок-схема.

При описании частей мобильного приложения использовалась диаграмма классов с подробным описанием классов и методов.

Для выполнения программирования методов, был использован язык программирования Java.

3 Программная реализация и тестирование

3.1 Программная реализация мобильного приложения

В ходе выполнения бакалаврской работы была задействована интегрированная среда разработки Android Studio.

Для реализации алгоритма сжатия для оперативной съемки и отправки проблемы для АХО ТГУ был выбран язык объектно-ориентированного программирования Java.

Для реализации модуля оперативной съемки было решено воспользоваться библиотекой fractal4j с открытым исходным кодом, которая включает фрактальный алгоритм сжатия изображения.

Функциональные возможности мобильного приложения:

- реализован вход в мобильное приложение с помощью логина и пароля или регистрации пользователя;
- реализована метод оперативной съемки с помощью открытия камеры устройства или выбор фото из галереи пользователя;
- реализована выбор темы проблемы и её категории с помощью ArrayList;
- реализован выбор адреса;
- реализована возможность более подробно описать проблему и подать запрос;
- реализовано отображения всех поданных запросов пользователя и их сортировка.

На рисунке 8 представлен проект разработанного мобильного приложения в интегрированной среде разработки Android Studio.

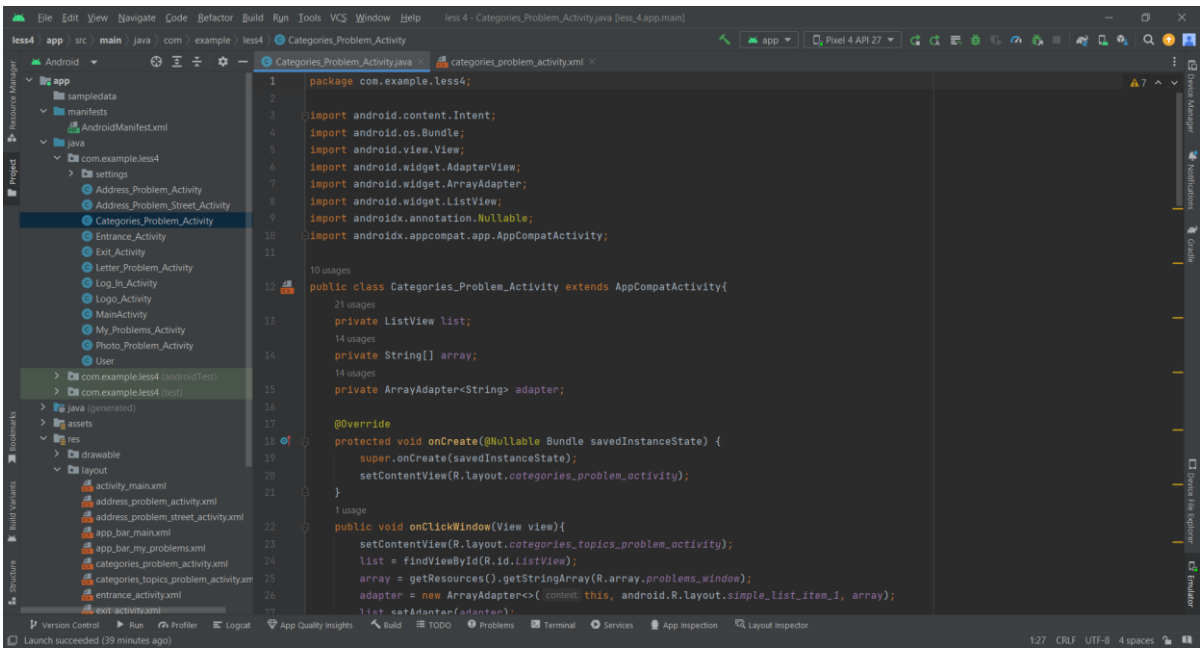

Рисунок 8 – Проект мобильного приложения в интегрированной среде разработки Android Studio

Для сохранения данных о пользователе мы будем использовать платформу Firebase. Это облачная база данных, позволяющая пользователям хранить и получать информацию.

Для контроля версий была выбрана платформа GitLab – это онлайн-сервис, который предоставляет полнофункциональную систему управления версиями Git, предназначенную для хранения, управления и совместной работы над кодом. Он также включает инструменты для автоматической непрерывной интеграции и развертывания, управления задачами и ошибками, а также функции управления доступом и безопасности.

На рисунке 9 представлен созданный репозиторий на платформе GitLab для реализации алгоритма сжатия для оперативной съемки и отправки проблемы для АХО ТГУ.

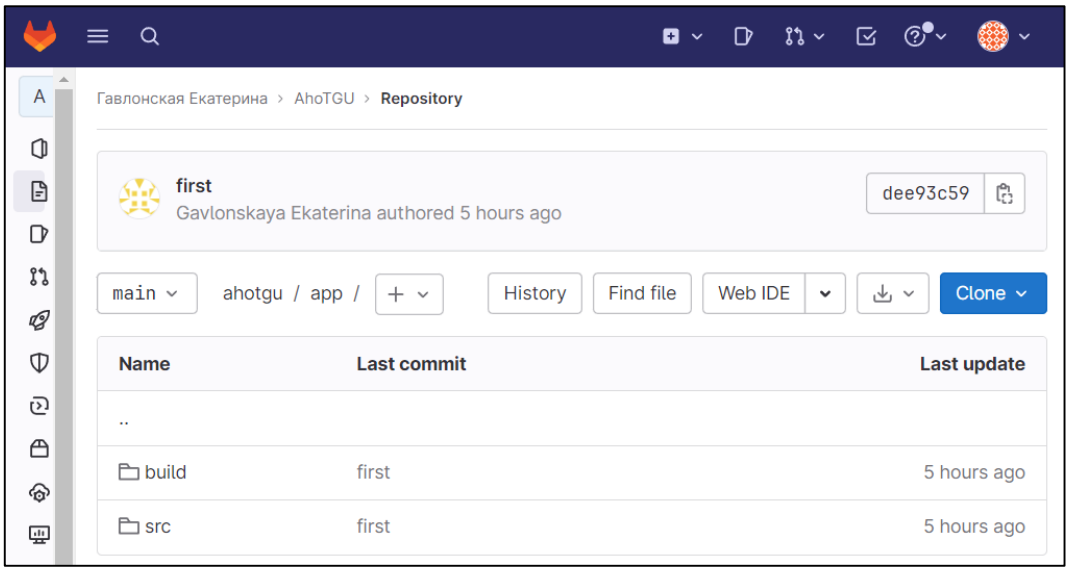

Рисунок 9 – Репозиторий для хранения версий мобильного приложения

Приступим к рассмотрению функционала мобильного приложения.

На рисунке 10 представлен фрагмент кода реализация метода авторизации пользователя в мобильном приложении.

```
public void onClickLogIn(View view) {
    Intent i = new Intent( packageContext: Entrance_Activity.this, Log_In_Activity.class);
    startActivity(i);
}
1 usage
public void onClickEntrance(View view) {
    if (edLogin.getText().toString().isEmpty() || edPassword.getText().toString().isEmpty()) {
        Toast.makeText( context: Entrance_Activity.this, text: "Заполните все поля", Toast.LENGTH_SHORT).show();
    } else {
        mAuth.signInWithEmailAndPassword(edLogin.getText().toString(), edPassword.getText().toString())
            .addOnCompleteListener(new OnCompleteListener<AuthResult>() {
                @Override
                public void onComplete(@NonNull Task<AuthResult> task) {
                    if (task.isSuccessful()) {
                        Intent i = new Intent( packageContext: Entrance_Activity.this, MainActivity.class);
                        startActivity(i);
                    } else {
                        Toast.makeText( context: Entrance_Activity.this, text: "Пользователь не найден", Toast.LENGTH_SHORT)
```

Рисунок 10 – Фрагмент кода реализация метода авторизации пользователя в мобильном приложении

На рисунке 11 представлен фрагмент кода реализует метод регистрации пользователя в мобильном приложении с помощью метода OnClickEntrance().

Для аутентификации используется метод mAuth.signInWithEmailAndPassword(). Если аутентификация прошла успешно, Firebase Authentication вернет объект FirebaseUser, представляющий аутентифицированного пользователя, и произойдет вход в приложение. В противном случае появится ошибка.

```
public void onClickLogIn(View view) {
    if (edNameRegister.getText().toString().isEmpty() ||
            edSurnameRegister.getText().toString().isEmpty() ||
            edLoginRegister.getText().toString().isEmpty() ||
            edPasswordRegister.getText().toString().isEmpty())
    {
        Toast.makeText( context: Log_In_Activity.this,  text: "Заполните все поля", Toast.LENGTH_SHORT).show();
    } else {
        mAuth.createUserWithEmailAndPassword(edLoginRegister.getText().toString(), edPasswordRegister.getText().toString())
            .addOnCompleteListener(new OnCompleteListener<AuthResult>() {
                @Override
                public void onComplete(@NonNull Task<AuthResult> task) {
                    if (task.isSuccessful()) {
                        String id = mDataBase.getKey();
                        String name = edNameRegister.getText().toString();
                        String surname = edSurnameRegister.getText().toString();
                        String login = edLoginRegister.getText().toString();
                        User newUser = new User(id, name, surname,login);
                        mDataBase.push().setValue(newUser);
                        Intent i = new Intent( packageContext: Log_In_Activity.this, Entrance_Activity.class);
                        startActivity(i);
                    } else {
                        Toast.makeText( context: Log_In_Activity.this,  text: "Возникла ошибка", Toast.LENGTH_SHORT).show();
                    }
```

Рисунок 11 – Фрагмент кода реализация метода регистрации пользователя в мобильном приложении

Метод mAuth.createUserWithEmailAndPassword, предоставляемый Firebase Authentication SDK, позволяет создавать нового пользователя в базе данных и аутентифицировать его.

После успешного создания нового пользователя в базе данных Firebase Authentication метод вернет объект FirebaseUser, представляющий созданного пользователя. Если в базе данных уже есть данный пользователь или некорректно введены логин или пароль, появится ошибка.

На рисунке 12 представлен фрагмент кода реализация метода фото проблемы в мобильном.

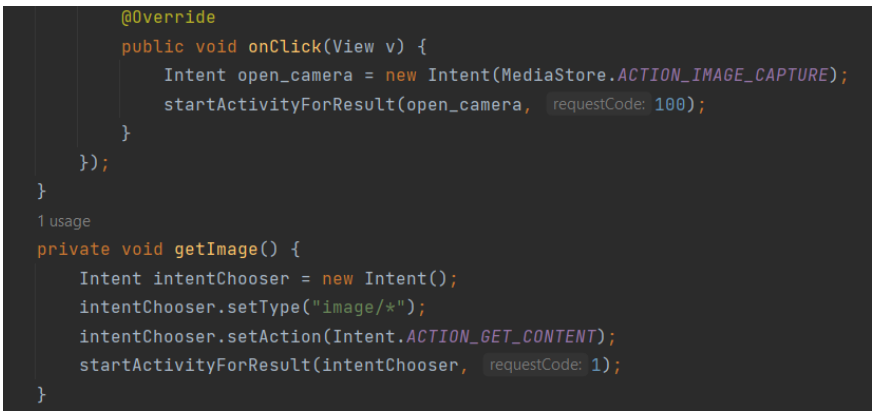

Рисунок 12 – Фрагмент кода реализация метода фото проблемы в мобильном приложении

Объект Intent с именем open_camera создается для запуска камеры на устройстве Android с помощью метода MediaStore.ACTION_IMAGE_CAPTURE.

Метод intentChooser.setType(«image/*») указывает, что необходимо открыть галерею с изображениями.

На рисунке 13 представлен фрагмент кода, где происходит выбор позволяющий выбрать между открытием галереи или запуском камеры.

Рисунок 13 – Фрагмент кода, где происходит выбор, позволяющий выбрать между открытием галереи или запуском камеры

Данный код является реализацией метода onActivityResult() в приложении для выбора изображений из галереи или для создания нового с помощью камеры.

На рисунке 14 представлен фрагмент кода реализация метода сжатия фотографии с помощью фрактального алгоритма.

```
imageView = findViewById(R.id.imageView);
Bitmap bitmap = BitmapFactory.decodeResource(getResources(), R.drawable.image_problem);
Bitmap compressedBitmap = FractalCompressionUtils.compress(bitmap);
imageView.setImageBitmap(compressedBitmap);
```

Рисунок 14 – Фрагмент кода реализация метода сжатия фотографии с помощью фрактального алгоритма

Fractal Compression – метод сжатия изображений, использующий математические алгоритмы и технологии фрактального анализа для замены повторяющихся узоров в изображении более компактными кодами. Результатом работы метода compress(), является сжатый объект Bitmap, который можно использовать в приложении для отображения изображения. Это позволяет уменьшить размер приложения и ускорить загрузку изображений, особенно на мобильных устройствах.

Однако, метод Fractal Compression может потребовать много времени и ресурсов для сжатия изображений с высоким разрешением или сложными текстурами, и повторное сжатие уже сжатого изображения может ухудшить качество. Поэтому перед использованием метода необходимо тщательно оценить требования к сжатию и выбрать наиболее подходящий метод сжатия для конкретного случая.

На рисунке 15 представлен фрагмент кода реализация метода категории проблемы в мобильном приложении.

Рисунок 15 – Фрагмент кода реализация метода категории проблемы в мобильном приложении

В методе OnClickWindow() нажатие на кнопку приводит к вызову ListView в макете пользовательского интерфейса, который привязывается к объекту list в коде активности

Метод getResources().getStringArray() является одним из наиболее удобных способов получения массива строк из ресурсов приложения в языке Java. Для получения массива строк с помощью этого метода необходимо передать ему идентификатор ресурса в качестве параметра. Возвращаемое значение будет массивом строк, содержащим строковые значения из этого ресурса.

ArrayAdapter – это класс, который обеспечивает связывание массива данных с ListView.

Метод setOnItemClickListener() вызывается на объекте ListView. Обработчик, установленный с помощью этого метода, будет вызываться при клике на элементе списка.

На рисунке 16 представлен фрагмент кода реализация метода адреса проблемы в мобильном приложении.

```
String[] countriesFrame = { "УЛК", "Э", "НИЧ", "Г", "У", "С", "В", "Д", "А", "Е", "Ф"};

@Override
protected void onCreate(@Nullable Bundle savedInstanceState){
    super.onCreate(savedInstanceState);
    setContentView(R.layout.address_problem_activity);
    Spinner spinnerFrame = findViewById(R.id.spinnerFram);
    ArrayAdapter<String> adapterFrame = new ArrayAdapter( context: this, android.R.layout.simple_spinner_item, countriesFrame);
    adapterFrame.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
    spinnerFrame.setAdapter(adapterFrame);
    AdapterView.OnItemSelectedListener itemSelectedListener = new AdapterView.OnItemSelectedListener() {
        @Override
        public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
            String item = (String)parent.getItemAtPosition(position);
        }
        @Override
        public void onNothingSelected(AdapterView<?> parent) {
        }
    };
    spinnerFrame.setOnItemSelectedListener(itemSelectedListener);
}
```

Рисунок 16 – Фрагмент кода реализация метода адреса проблемы в мобильном приложении

На фрагменте кода String[] countriesFrame – это объявление переменной массива строк (String), где идет перечисление корпусов ТГУ.

Spinner – это элемент пользовательского интерфейса, который позволяет выбирать один из нескольких вариантов, используя выпадающий список. Этот элемент очень удобен для пользователей, когда необходимо выбрать из нескольких вариантов, например, для выбора валюты или страны в приложении. Для работы с Spinner в коде приложения можно использовать класс Spinner, который является частью Android SDK. В переменной spinnerFrame можно хранить ссылку на объект Spinner, который был создан в коде приложения, и использовать его для дальнейшей работы с элементом в приложении.

setDropDownViewResource – это метод, позволяющий задать представление для каждого элемента списка при его отображении в выпадающем списке.

На рисунке 17 представлен фрагмент кода реализация метода описания проблемы в мобильном приложении.

Рисунок 17 – Фрагмент кода реализация метода описания проблемы в мобильном приложении

На фрагменте кода FirebaseDatabase.getInstance() – это метод, который возвращает экземпляр Firebase Realtime Database, связанный с текущим приложением. Он позволяет программистам получать доступ к базе данных Firebase, позволяя им сохранять и извлекать данные из базы данных в реальном времени.

getReference() – это метод, который возвращает ссылку на корневой узел базы данных.

child() – это метод, который позволяет указать имя дочернего узла для текущего узла базы данных. Он позволяет производить операции чтения и записи данных в реальном времени.

setValue() – это метод, который используется для записи данных в Firebase Realtime Database.

File file = new File(getFilesDir(), FILE_NAME) – это создание объекта типа File, представляющего файл на внутреннем хранилище приложения.

В результате реализации мобильного приложения получается на выходе рабочие методы которых формируют мобильное приложение с возможностью оперативной съемки и отправки проблемы для АХО ТГУ. Приложение оснащено функцией оперативной съемки и отправки данных, что увеличивает его удобство и функциональность.

3.2 Тестирование мобильного приложения

Результатом тестирования системы должно стать точное соответствие поставленной задаче. Точность выполнения задачи будет проверена с помощью тестирования системы. Для этого мы будем использовать изображения из мобильного приложения, а также камеру с разрешением 48 Мп (1080 x 2400 px). Для проверки отправки запроса мы воспользуемся облачной базой данных Firebase.

На рисунке 18 представлен процесс входа в мобильное приложение. Для этого произведем регистрацию, а затем войдем под своими данными в мобильное приложение.

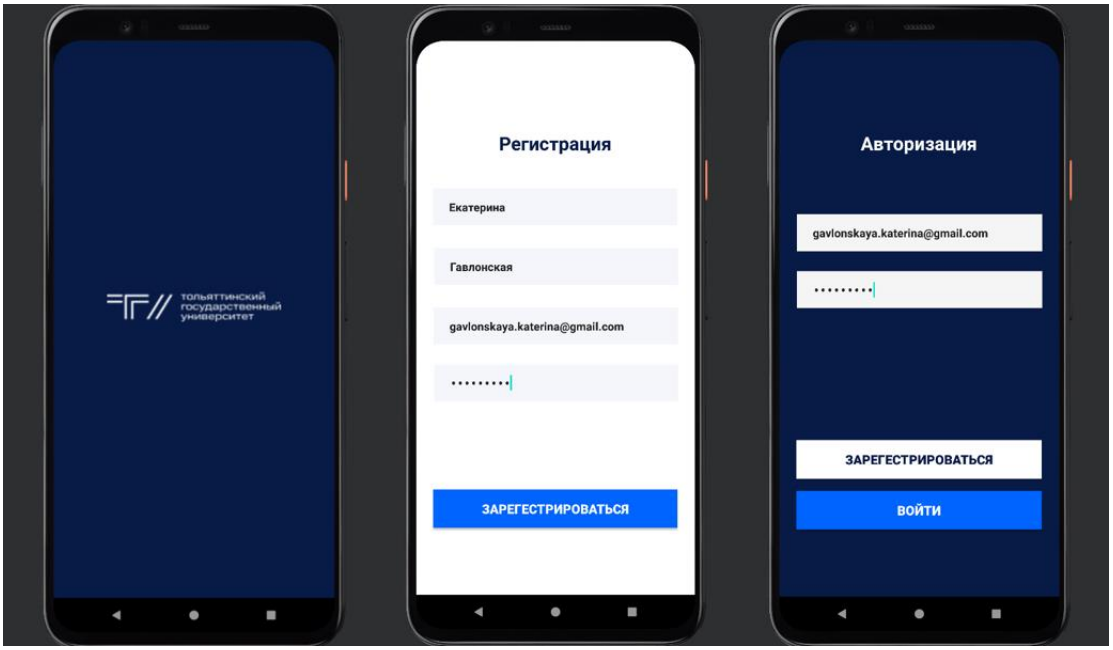

Рисунок 18 – Вход пользователя в мобильное приложение

На рисунке 19 представлен процесс перехода пользователя в раздел «Сообщить о проблеме». Первым этапом нужно сделать фото проблемы или загрузить фото из галереи мобильного устройства.

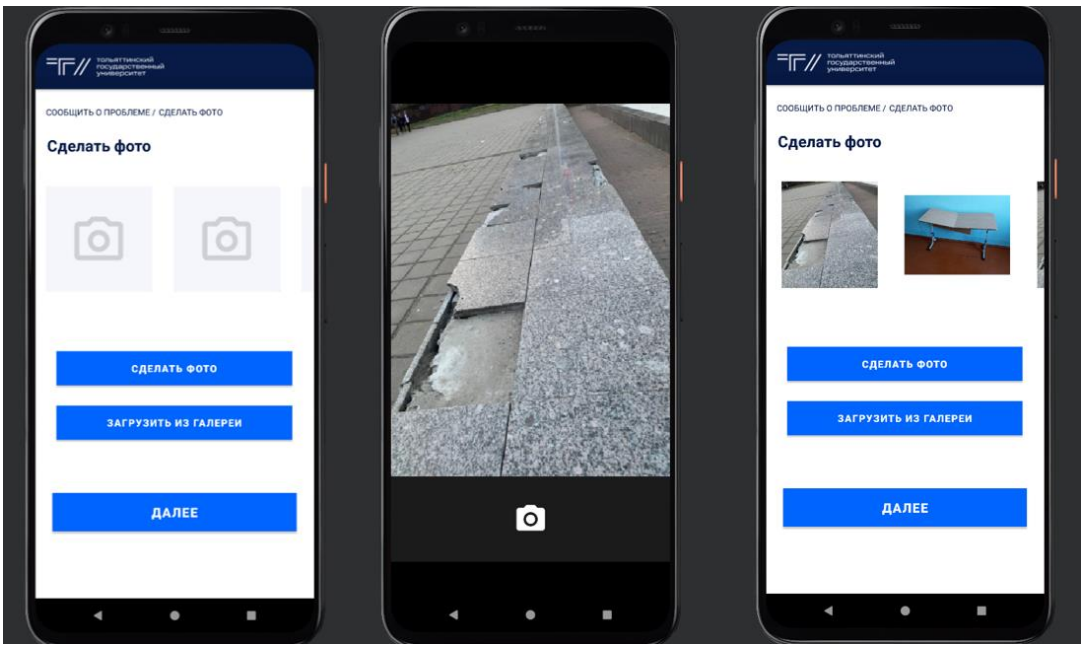

Рисунок 19 – Первый этап «Сделать фото» в мобильном приложении

На рисунке 20 представлен процесс перехода пользователя на второй этап подачи запроса, где ему необходимо выбрать категорию и тему проблемы.

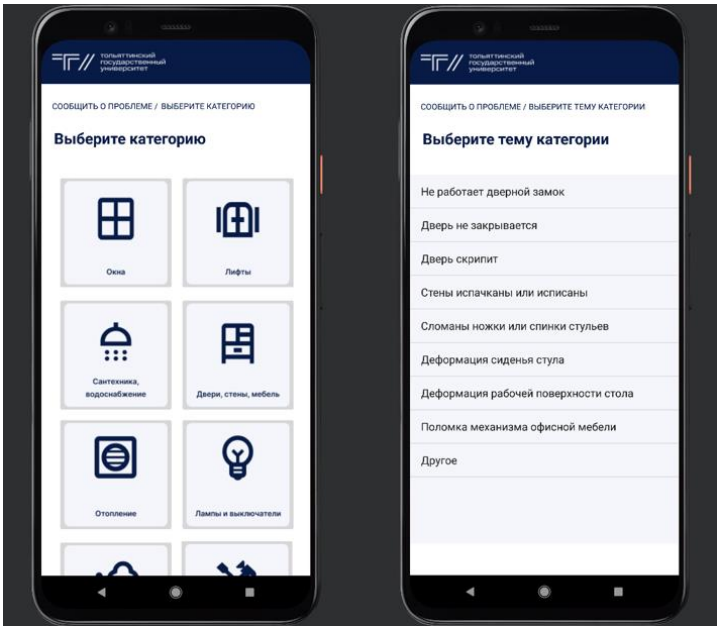

Рисунок 20 – Второй этап «Выбрать категорию» и «Выбрать тему категории» в мобильном приложении

На рисунке 21 представлен процесс перехода пользователя на третий этап, где указывает адрес проблемы. После ввода адреса, пользователь переходит на четвертый этап, где добавляет дополнительное описание проблемы и нажимает кнопку «Подать запрос» для отправки запроса.

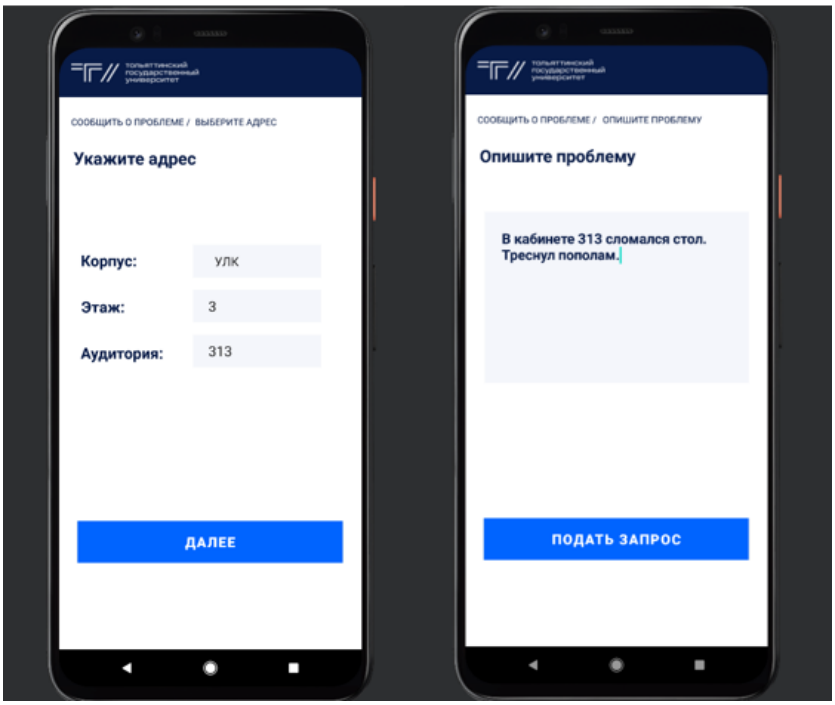

Рисунок 21 – Этапы «Укажите адрес» и «Опишите проблему» в мобильном приложении

На рисунке 22 представлено тестирование мобильного приложения на наличие отправленного нами запроса. В качестве проверки будем использовать платформу Firebase Realtime Database, где будем получать данные о запросе.

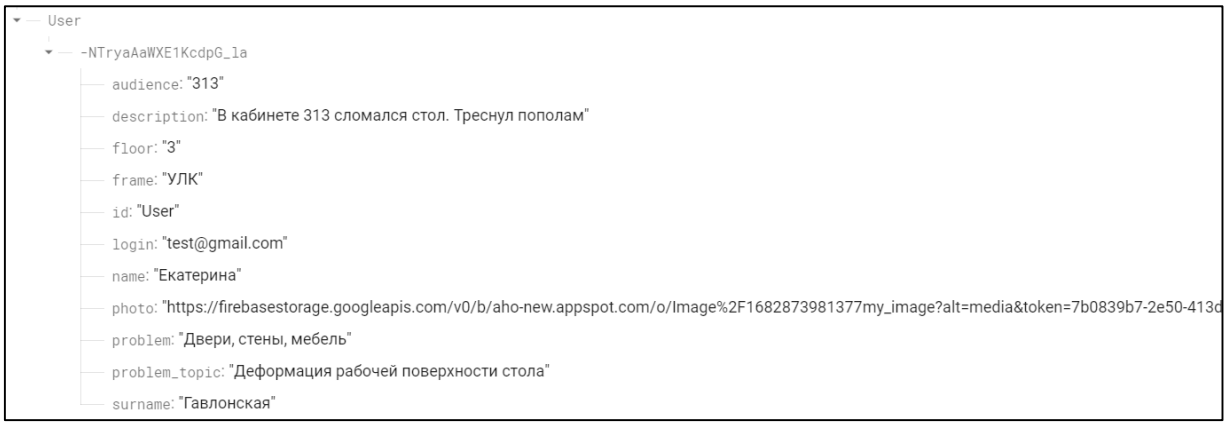

Рисунок 22 – Тестирование мобильного приложения

Таким образом, в результате проведенных работ было разработан алгоритм сжатия для оперативной съемки и отправки проблемы для АХО ТГУ. После успешного тестирования можно сделать вывод, что цель работы была достигнута и разработанное приложение готово к использованию.

Выводы по третьему разделу

В разделе, посвященном программной реализации и тестированию мобильного приложения, были проведены соответствующие тесты. Результаты тестирования подтвердили, что программа успешно справилась со всеми задачами, которые были перед ней поставлены. Это свидетельствует о том, что мобильное приложение работает правильно и готово к использованию.

Заключение

В выполненной выпускной квалификационной работе были изучены и рассмотрены основные аспекты, связанные с разработкой мобильного приложения.

При проведении исследования предметной области были представлены материалы, содержащие информацию о теоретических аспектах и практической реализации алгоритмов сжатия изображений.

При описании программной реализации были исследованы и применены алгоритмы сжатия изображений, а также были реализованы модули программы на языке Java в интегрированной среде разработки Android Studio, специальная версия IntelliJ IDEA, которая включает в себя версию Android SDK и дополнительные инструменты графических интерфейсов, облегчающие разработку приложений.

В процессе тестирования модулей были предоставлены примеры входных данных и их взаимодействия, которые демонстрировали правильность работы всех алгоритмов. В результате тестирования не было обнаружено ошибок, а полученные результаты при взаимодействии с программой соответствовали ожиданиям. Исходя из всей предоставленной информации, можно заключить, что данное мобильное приложение работает безупречно и корректно.

Подводя итог всем представленным данным, можно утверждать, что для успешной разработки данного мобильного приложения необходимы глубокие знания и опыт в области мобильной разработки, включая работу с Android Studio и алгоритмами сжатия изображений, а также владение языком программирования Java.

Таким образом, в результате проделанной работы было разработан алгоритм сжатия для оперативной съемки и отправки проблемы для АХО ТГУ.

Список используемой литературы и используемых источников

1. Ахметов А. К. Операционная система Android: история создания и развития. Разработка приложений для платформы Android [Электронный ресурс]. – Режим доступа: https://cyberleninka.ru/article/n/operatsionnayasistema-android-istoriya-sozdaniya-i-razvitiya-razrabotka-prilozheniy-dlya-plat formy-android (дата обращения: 04.02.2023).

2. Ватолин Д., Ратушняк А., Смирнов М., Юкин В. Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео. - М.: ДИАЛОГ-МИФИ, 2003 - 384 с.

3. Введение в разработку приложений для смартфонов на ОС Android [Электронный ресурс]. – Режим доступа: https://intuit.ru/studies/professional_skill_improvements/ (дата обращения: 10.03.2023).

4. Глушенко, С. А., Долженко, А. И. Разработка мобильных приложений: Учебное пособие – Ростов-на-Дону: издательство РГЭУ (РИНХ), 2018 – 221 с.

5. ГОСТ Р 2.105-2019. Национальный стандарт Российской Федерации. Единая система конструкторской документации. Общие требования к текстовым документам. Введ. 2020-02-01.- М.: Изд-во стандартов, 2020. 36 с.

6. ГОСТ 2.316-2008. Межгосударственный стандарт. Единая система конструкторской документации. Правила нанесения надписей, технических требований и таблиц на графических документах. Общие положения. Взамен ГОСТ 2.316-68; Введ. 2009-07-01.- М.: Изд-во стандартов, 2009. 12 с.

7. ГОСТ 7.9-95. Межгосударственный стандарт. Система стандартов по информации, библиотечному и издательскому делу. Реферат и аннотация. Общие требования (ИСО 214-76). Взамен ГОСТ 7.9-77; Введ. 30.06.1997.- М.: Изд-во стандартов, 2001. 28 с.

8. ГОСТ Р 54521-2011. Национальный стандарт Российской

Федерации. Статистические методы. Математические символы и знаки для применения в стандартах (ИСО 80000-2:2009). Введ. 2012-12-01.- М.: Изд-во стандартов, 2011. 32 с.

9. Гриффитс Дон, Гриффитс Дэвид. Программирование для Android. — СПб.: Питер, 2016 — 704 с.: ил. – (Серия «Head First O'Reilly»).

10. Дарвин, Ян Ф. Д20 Android. Сборник рецептов: задачи и решения для разработчиков приложений, 2-е изд.: Пер. с англ. - СпБ.: ООО "Альфа книга", 2018. - 768 с.

11. Дейтел П., Дейтел Х., Уолд А. Д27 Android для разработчиков / 3-е изд. — СПб.: Питер, 2016. — 512 с.

12. Детальный анализ Android [Электронный ресурс]. – Режим доступа: https://xakep.ru/2014/07/03/art-vm/ (дата обращения: 19.05.2023).

13. Д. Сэломон. Сжатие данных, изображений и звука. Москва: Техносфера, 2004. - 368с.

14. Майер Р. Android 4. Программирование приложений для планшетных компьютеров и смартфонов / Рето Майер ; [пер. с англ. ООО «Айдио-номикс»]. - М. : Эксмо, 2013. - 816 с. - (Мировой компьютерный бестселлер).

15. Мандельброт Б. Фрактальная геометрия природы. – Москва: Институт компьютерных исследований, 2002, 656 с.

16. Отношения и их графическое изображение на диаграмме классов [Электронный ресурс]. – Режим доступа: https://intuit.ru/studies/courses/32/32/lecture/ (дата обращения: 07.03.2023).

17. Разработка мобильных приложений. Первые шаги / М. А. Федотенко; под ред. В. В. Тарапаты. – М. : Лаборатория знаний, 2019.—335 с.

18. С. Уэлетид. Фракталы и вейвлеты для сжатия изображений в действии. Учебное пособ. - М.: Издательство Триумф, 2003 - 320 с.

19. Тропченко А.Ю., Тропченко А.А. Методы сжатия изображений, аудиосигналов и видео: Учебное пособие – СПб: СПбГУ ИТМО, 2009. – 108 с.

20. Филлипс Б., Стюарт К., Марсикано К. Android. Программирование

для профессионалов. 3-е изд. - СПб.: Питер, 2017. - 688 с.: ил. — (Серия «Для профессионалов»).

21. Adam Gerber, Clifton Craig, David Selvaraj – Learn Android Studio Build Android Apps Quickly and Effectively, 2015. 465pages.

22. John Wiley & Sons – Beginning Android Programming with Android Studio, 2017. 435 pages.

23. Salomon, D. (David), 1938 – Data compression / David Salomon. p. cm. Data compression (Computer science), 2004. 899 pages.

24. Sayood, Khalid. Introduction to data compression/ Khalid Sayood.—3rd ed. p. cm. Includes bib, 2000. 680 pages.

25. Wayner P. Compression Algorithms for Real Programmers, 1999. 240 pages.