ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ
Северо-Кавказский филиал
ордена Трудового Красного Знамени федерального государственного бюджетного образовательного учреждения высшего образования
"Московский технический университет связи и информатики"

Методические указания
к лабораторным и практическим занятиям

СЖАТИЕ И ХРАНЕНИЕ ИНФОРМАЦИИ

Направление подготовки:
09.03.01 Информатика и вычислительная техника

Ростов-на-Дону
2019

УДК 681.3.06 (076)
ББК 32.07

Чикалов А.Н. Сжатие и хранение информации. Методические указания к практическим занятиям. Ростов-на-Дону: Северо-Кавказский филиал МТУСИ, 2019.- 31 с.

В пособии изложены методические рекомендации, содержательные материалы и контрольные задания для проведения практических занятий по изучению алгоритмических основ сжатия информации, применении их для обработки данных, аудио и видео информации, использовании информационных технологий сжатия информации.

Методические указания предназначены для студентов, обучающихся по направлениям подготовки 09.03.01 Информатика и вычислительная техника профилей Вычислительные машины, комплексы, системы и сети, Программное обеспечение и интеллектуальные системы.

Пособие предназначено для использования при изучении дисциплины Сжатие и хранение информации, а также может быть использовано преподавателями и студентами при изучении родственных дисциплин или разделов и в процессе самостоятельной работы.

Учебное пособие обсуждено и одобрено на заседании кафедры ИВТ
Протокол от 26 августа 2019 г. № 1

Рецензент Зав. кафедрой ИВТ д.т.н. профессор Соколов С.В.

СОДЕРЖАНИЕ

1. ИЗУЧЕНИЕ АЛГОРИТМА ХАФФМАНА

Цель

1. Изучить принципы реализации алгоритма Хаффмана, его характеристики и ограничения;

2. Совершенствовать навыки использования языков и сред программирования для реализации алгоритмов сжатия;

3. Совершенствовать навыки анализа, обобщения и систематизации полученных результатов, навыки составления и оформления отчетных материалов, навыки точного и лаконичного представления докладов на вопросы технического характера.

Учебные вопросы

1. Разработка словесного описания алгоритма, схемы алгоритма;
2. Подготовка исходного текста программы;
3. Отладка и оценка работоспособности программы сжатия.

Литература для подготовки к занятию

1. Ватолин Д. и др. Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео. - М.: ДИАЛОГ-МИФИ, 2003. - 384 с.
2. Тропченко А.Ю., Тропченко А.А. Методы сжатия изображений, аудиосигналов и видео: Учебное пособие. - СПб.: СПбГУ ИТМО, 2009. - 108 с.

Содержание отчета

1. Название работы;

2. Для каждого задания: название задания и материал в объеме, указанном в задании.

Одним из наиболее широко распространенных видов сервисных программ являются программы, предназначенные для архивации, упаковки файлов путем сжатия хранимой в них информации.

Сжатие информации — это процесс преобразования информации, хранящейся в файле, к виду, при котором уменьшается избыточность в ее представлении и соответственно требуется меньший объем памяти для хранения.

Сжатие информации в файлах производится за счет устранения избыточности различными способами, например за счет упрощения кодов, исключения из них постоянных битов или представления повторяющихся символов или повторяющейся последовательности символов в виде коэффициента повторения и соответствующих символов.

Цель процесса сжатия – получение более компактного выходного потока информационных единиц из некоторого изначально некомпактного входного потока при помощи некоторого их преобразования.

Основными техническими характеристиками процессов сжатия и результатов их работы являются:

- **степень сжатия** (compress rating) или отношение (ratio) объемов исходного и результирующего потоков;
- **скорость сжатия** - время, затрачиваемое на сжатие некоторого объема информации входного потока, до получения из него эквивалентного выходного потока;
- **качество сжатия** - величина, показывающая на сколько сильно упакован выходной поток, при помощи применения к нему повторного сжатия по этому же или иному алгоритму.

Все способы сжатия можно разделить на две категории: обратимое и необратимое сжатие.

Под **необратимым** сжатием подразумевают такое преобразование входного потока данных, при котором выходной поток, основанный на определенном формате информации, представляет, с некоторой точки зрения, объект, достаточно похожий по внешним характеристикам на входной поток, однако отличается от него объемом. Данный подход реализован в популярных форматах представления видео и фото информации, известных как JPEG и JFEF алгоритмы и JPG и GIF форматы файлов. Необратимое сжатие невозможно применять в областях, в которых необходимо иметь точное соответствие информационной структуры входного и выходного потоков.

Обратимое сжатие всегда приводит к снижению объема выходного потока информации без изменения его информативности, т.е. без потери информационной структуры. Более того, из выходного потока, при помощи восстанавливающего или декомпрессирующего алгоритма, можно получить входной, а процесс восстановления называется декомпрессией или распаковкой и только после процесса распаковки данные пригодны для обработки в соответствии с их внутренним форматом

Первые теоретические разработки в области сжатия информации относятся к концу 40-х годов. В конце семидесятых появились работы Шеннона, Фано и Хафмана. К этому времени относится и создание алгоритма FGK (Faller, Gallager, Knuth), где используется идея "сродства", а получатель и отправитель динамически меняют дерево кодов. Пропускная способность каналов связи более дорогостоящий ресурс, чем дисковое пространство, по этой причине сжатие данных до или во время их передачи еще более актуально.

Здесь целью сжатия информации является экономия пропускной способности и в конечном итоге ее увеличение. Все известные алгоритмы сжатия сводятся к шифрованию входной информации, а принимающая сторона выполняет дешифровку принятых данных.

Сжатие информации без потерь осуществляется статистическим кодированием или на основе предварительно созданного словаря. Статистические алгоритмы (напр., схема кодирования Хаффмана) присваивают каждому

входному символу определенный код. При этом наиболее часто используемому символу присваивается наиболее короткий код, а наиболее редкому - более длинный. Таблицы кодирования создаются заранее и имеют ограниченный размер. Этот алгоритм обеспечивает наибольшее быстродействие и наименьшие задержки. Для получения высоких коэффициентов сжатия статистический метод требует больших объемов памяти.

Задание 1.1 Разработка словесного описания алгоритма, схемы алгоритма

Сжатие файла по алгоритму Хаффмана основывается на частоте повторения символов. Для сжатия необходимо подсчитать сколько раз встречается каждый символ из расширенного набора ASCII. После подсчета частоты вхождения каждого символа, необходимо отсортировать получившуюся таблицу (символ – частота вхождения) по частоте. Каждую ссылку из таблицы назовем "узлом". В дальнейшем (в дереве) мы будем позже размещать указатели, которые будут указывать на этот "узел".

На вход программы подаётся файл, содержащий текст. Затем осуществляется анализ текста, выполняется подсчёт частот находящихся в нём различных символов, заполнение массивов данных и их сортировка.

Следующий шаг -построение бинарного дерева. Элементы, имеющие большие частоты, располагаются справа, а элементы с меньшими частотами - слева.

При построении кода для каждого символа программа проходит по левым ссылкам узлов дерева до первого встреченного листа и обнаруживает первый элемент для объединения. После этого лист удаляется из дерева, и начинается аналогичный проход для поиска второго элемента, который после обнаружения также удаляется из дерева.

К коду всех символов первого из найденных листов добавляется единица, к коду всех символов второго листа добавляется ноль. Так, например, если символ «а» имел код 01, то, если он окажется в первом из найденных листов, код примет вид 101, если же во втором, то 001.

После нахождения двух элементов происходит их объединение в один новый элемент, причем частоты обоих суммируются, и получившееся значение присваивается новому элементу. Для нахождения места расположения образованного элемента в дереве сравниваются частоты слева и справа от текущей позиции, начиная с корневого узла. На основании этих сравнений выполняется передвижение по дереву либо влево, либо в право. Если находится узел, частота которого равна частоте образованного элемента, происходит его вставка в текущую позицию, а найденный узел прикрепляется к вставленному слева.

Возможен случай, когда частота образованного элемента больше, чем частота узла, стоящего справа относительно текущей позиции. Если при этом выполнить действия, аналогичные действиям выше, и вставить новый элемент в дерево на место текущего, то возможна ошибка обхода дерева, в результате чего образованные коды символов будут некорректны. Для того чтобы это исправить запускается алгоритм, перестраивающий бинарное дерево. После подобной перестройки алгоритм образования новых узлов повторяется.

Программа продолжает свою работу до тех пор, пока в дереве не останется один элемент. Это означает, что все элементы были обработаны и для всех символов, которые располагались в исходном файле, были построены коды.

После этого выводится исходная строка и количество бит, занимаемых этой строкой, закодированная строка и количество бит, которые требуются для ее кодирования.

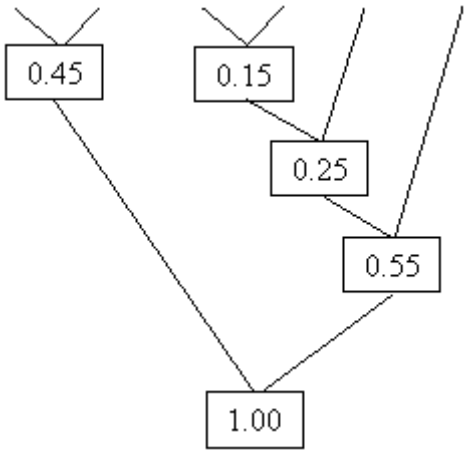
В качестве входных можно использовать тексты, содержащие символы русского и английского алфавитов, фрагменты кода программ и другие символы.

Пример:

Допустим, исходный файл длинной в 100 байт и состоит из 6 различных символов. Частота вхождения каждого из символов в файл представлена в таблице:

A	B	C	D	E	F
0.25	0.2	0.05	0.1	0.1	0.3

В дальнейшем элементы таблицы считаются узлами, а их вероятность – весом узлов. Дерево формируется следующим образом. Дальнейший порядок следующий.



1. Элементы таблицы сортируются в порядке убывания их частот.

2. Сначала выбираются два узла с наименьшим весом (два последних элемента), вместо них создаётся новый узел (Вершина #n), с весом, равным их сумме. Новая вершина становится элементом таблицы вместо использованных двух узлов и занимает свое место в порядке убывания весов.

3. Повторяется данная операция шага 2 до тех пор, пока не останется один узел с весом в единицу.

4. Далее выполняется кодирование. Для этого выполняется обход дерева от корня (узла с весом в 1) и спуск до самих символов (узлов, листьев дерева). При этом прослеживаются вверх по дереву все повороты ветвей и, если выполняется левый поворот, то запоминается 0-й бит, если правый - 1-й бит.

Таким образом, для символа A получается код 00, аналогично для других:

A	B	C	D	E	F
00	01	1000	1001	101	11

Подсчет сжатия показывает:

было 100 байт = 800 бит

стало 25*2 + 20*2 + 5*4 + 10*4 + 10*3 + 30*2 = 240 бит = 30 байт

Получается, что сжатие на 70%.

Для разархивирования необходимо сохранить дерево либо непосредственно таблица частот, и тогда при разархивировании потребуется дерево восстановить заново.

Для сохранения таблицы частот потребуется:

6 символов = 6 байт;

6 частот = 6*4 = 24 байт, итого 30+6+24 = 60 байт.

Также обычно сохраняют служебную информацию (время создания, время редактирования, атрибуты, имя файла): 4+4+1+11 (длину файла считают 11 символов, включая расширение). Всего 20 байт.

Получается архив в 20+60 = 80 байт.

Окончательно сжатие на 20%. В среднем алгоритм Хаффмана даёт 20-25% сжатие (для текстовых документов).

При этом нет необходимости в байтах-флагах, т. е. не надо разделять битовые серии.

Недостатком следует признать необходимость двух проходов во время архивирования: первый – подсчёт количества вхождений каждого байта, второй – непосредственно архивирование.

В отчете представить:

1.Схему алгоритма;

2.Описание переменных.

Задание 1.2. Подготовка исходного текста программы

В соответствии с разработанным алгоритмом составить программу его реализации на одном из языков программирования. Использовать освоенные среды программирования.

В отчете представить:

1.Исходный текст программы.

2.Комментарии к созданным блокам и процедурам.

Задание1.3.Отладка и оценка работоспособности программы сжатия

Провести отладку программы на примере подготовленного тестового файла для сжатия.

Выполнить тестовые прогоны для оценки параметров сжатия.

В отчете представить:

1.Отлаженную программу сжатия;

2. Данные тестовых прогонов на различных файлах (время работы, степень сжатия, влияние уникальных символов на степень сжатия).

Вопросы для самопроверки

1. В чем смысл понятия: избыточность сообщения?
2. Что определяет коэффициент избыточности?
3. Алгоритм построения дерева Хаффмана.
4. Какие задачи решает кодирование методом Хаффмана?
5. При кодировании каких данных можно использовать сжатие данных с потерями? Ответ обоснуйте.
6. В чем преимущества и недостатки статических методов и словарного сжатия?
7. Каким образом кодирование по алгоритму Хаффмана через префиксный код гарантирует минимальную длину кода?
8. За счет чего в методе Хаффмана поддерживается однозначность соответствия кода кодируемому символу?
9. Почему алгоритм Хаффмана малоэффективен для файлов маленьких размеров?
10. Выполните кодирование по методу Хаффмана через префиксный код символов, которые встречаются с вероятностями 0,3; 0,2; 0,1; 0,1; 0,1; 0,05; 0,05; 0,04; 0,03; 0,03. Сравните полученный результат с данными программной реализации.
11. Докажите, что метод Хаффмана кодирует информацию без потерь.
12. Порядок работы алгоритма Хаффмана
13. Построение оптимального кодового дерева.
14. Средняя длина кода и ее расчет.

2. ИЗУЧЕНИЕ АЛГОРИТМА LZ

Цель

1. Изучить принципы реализации алгоритма Лемпеля-Зива, его характеристики и ограничения;

2. Совершенствовать навыки использования языков и сред программирования для реализации алгоритмов сжатия;

3. Совершенствовать навыки анализа, обобщения и систематизации полученных результатов, навыки составления и оформления отчетных материалов, навыки точного и лаконичного представления докладов на вопросы технического характера.

Учебные вопросы

1. Разработка словесного описания алгоритма, схемы алгоритма;
2. Подготовка исходного текста программы;
3. Отладка и оценка работоспособности программы сжатия.

Литература для подготовки к занятию

1. Ватолин Д. и др. Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео. - М.: ДИАЛОГ-МИФИ, 2003. - 384 с.
2. Тропченко А.Ю., Тропченко А.А. Методы сжатия изображений, аудиосигналов и видео: Учебное пособие. - СПб.: СПбГУ ИТМО, 2009. - 108 с.

Содержание отчета

1. Название работы;

2. Для каждого задания: название задания и материал в объеме, указанном в задании.

Алгоритм RLE (Run Lenght Encoding - кодирование длин повторов) предполагает представление изображения в виде последовательности байтов по строкам растра. Сжатие происходит повторяющихся байтов путем замены их парами "Счетчик повторений, значение байта".

Признаком счетчика служат единицы двух старших бит в байте. Оставшиеся 6 бит указывают количество повторений значения второго байта этой пары:

1-й байт		2-й байт
11	значение счетчика	Код

Алгоритм рассчитан на деловую графику - изображения с большими областями повторяющегося цвета. В общем случае возможны файлы, для которых сжатый файл может быть больше исходного.

В 1977 году Абрахам Лемпель и Якоб Зив предложили алгоритм сжатия данных, названный позднее LZ77. Этот алгоритм используется в программах архивирования текстов compress, lha, pkzip и arj. Модификация алгоритма LZ78 применяется для сжатия двоичных данных. В основе лежит метод LZW (по символам имен разработчиков Lempel, Ziv и Welch), предполагающий сжатие за счет кодирования одинаковых цепочек байтов.

Эти модификации алгоритма защищены патентами США. Алгоритм предполагает кодирование последовательности бит путем разбивки ее на фразы с последующим кодированием этих фраз. Суть алгоритма заключается в следующем.

Если в тексте встретится повторение строк символов, то повторные строки заменяются ссылками (указателями) на исходную строку. Ссылка имеет формат <префикс, расстояние, длина>. Префикс в этом случае равен 1. Поле расстояние идентифицирует слово в словаре строк. Если строки в словаре нет, генерируется код символ вида <префикс, символ>, где поле префикс = 0, а поле символ соответствует текущему символу исходного текста. Отсюда видно, что префикс служит для разделения кодов указателя от кодов символ. Введение кодов <символ> позволяет оптимизировать словарь и поднять эффективность сжатия. Главная алгоритмическая проблема здесь заключается в оптимальном выборе строк, так как это предполагает значительный объем переборов.

Альтернативой статистическому алгоритму стала схема сжатия, основанная на динамически изменяемом словаре (напр., алгоритмы Лемпеля-Зива). Данный метод предполагает замену потока символов кодами, записанными в памяти в виде словаря (таблица перекодировки). Соотношение между символами и кодами меняется вместе с изменением данных. Таблицы кодирования периодически меняются, что делает метод более гибким. Размер небольших словарей лежит в пределах 2-32 килобайт, но более высоких коэффициентов сжатия можно достичь при заметно большими словарями до 400 килобайт.

Реализация алгоритма возможна в двух режимах: непрерывном и пакетном. Первый использует для создания и поддержки словаря непрерывный поток символов. При этом возможен многопротокольный режим (например, TCP/IP и DECnet). Словари сжатия и декомпрессии должны изменяться синхронно, а канал должен быть достаточно надежен (напр., X.25 или PPP), что гарантирует отсутствие искажения словаря при повреждении или потере пакета. При искажении одного из словарей оба ликвидируются и должны быть созданы вновь.

Пакетный режим сжатия также использует поток символов для создания и поддержания словаря, но поток здесь ограничен одним пакетом и по этой причине синхронизация словарей ограничена границами кадра. Для па-

кетного режима достаточно иметь словарь объемом, порядка 4 Кбайт. Непрерывный режим обеспечивает лучшие коэффициенты сжатия, но задержка получения информации (сумма времен сжатия и декомпрессии) при этом больше, чем в пакетном режиме.

Задание 2.1. Разработка словесного описания алгоритма, схемы алгоритма

Процесс сжатия выглядит достаточно просто. Считываются последовательно символы входного потока и проверяются, есть ли в созданной таблице словаря такая строка. Если строка есть, то считывается следующий символ, а если строки нет, то заносится в поток код для предыдущей найденной строки, заносится строка в таблицу и начинается поиск снова.

Инициализированная таблица словаря должна содержать все возможные строки, состоящие из одного символа. При сжатии байтовых данных в таблице будет 256 строк, а номер строки будет ее кодом. Для кода очистки и кода конца информации зарезервированы значения 256 и 257. Для 12-битового кода кодировки для строк еще остается довольно много значений.

Например, пусть сжимается последовательность 45, 55, 55, 151, 55, 55, 55. Тогда, согласно алгоритму, помещается в выходной поток сначала код очистки <256>, потом добавляется к изначально пустой строке "45" и проверяется, есть ли строка "45" в таблице. Поскольку при инициализации занесены в таблицу все строки длиной в один символ, то строка "45" есть в таблице. Далее читается следующий символ 55 из входного потока и проверяется, есть ли строка "45, 55" в таблице. Такой строки в таблице пока нет. В таблицу заносится строка "45, 55" (с первым свободным кодом 258) и записывается в поток код <45>. Можно коротко представить архивацию так:

"45" - есть в таблице;

"45, 55" - нет. Добавляется в таблицу <258>"45, 55". В поток: <45>;

"55, 55" - нет. В таблицу: <259>"55,55". В поток: <55>;

"55,151" - нет. В таблицу: <260>"55, 151". В поток: <55>;

"151,55" - нет. В таблицу: <261>"151,55". В поток: <151>;

"55, 55" - есть в таблице;

"55,55,55" - нет. В таблицу: "55,55, 55" <262>. В поток: <259>.

Последовательность кодов для данного примера, попадающих в выходной поток: <256>, <45>, <55>, <55>, <151>, <259>.

Особенность LZW заключается в том, что для декомпрессии не надо сохранять таблицу строк в файле для распаковки. Алгоритм построен таким образом, что есть возможность восстановить таблицу строк, пользуясь только потоком кодов.

Для каждого кода надо добавлять в таблицу строку, состоящую из уже присутствующей там строки и символа, с которого начинается следующая строка в потоке.

В отчете представить:

1. Схему алгоритма процедуры сжатия;
2. Имена и типы использованных переменных.

Задание 2.2. Подготовка исходного текста программы

В соответствии с разработанным алгоритмом составить программу его реализации на одном из языков программирования. Использовать освоенные среды программирования.

В отчете представить:

1.Исходный текст программы.

2.Комментарии к созданным блокам и процедурам.

Задание 2.3. Отладка и оценка работоспособности программы сжатия

Провести отладку программы на примере подготовленного тестового (или иного) файла для сжатия.

Выполнить тестовые прогоны для оценки параметров сжатия.

В отчете представить:

1.Отлаженную программу сжатия;

2. Данные тестовых прогонов на различных файлах (время работы, степень сжатия).

Вопросы для самопроверки

1. В чем состоит основной принцип работы алгоритма LZ77?
2. В чем смысл принципа скользящего окна?
3. В чем состоит основной принцип работы алгоритма LZ78?
4. Основное отличие алгоритма LZ78 от алгоритма LZ77?
5. Приведите определение процесса сжатия данных.
6. Приведите определение неискажающего сжатия цифровых данных (сжатие без потерь).
7. Приведите определение сжатия цифровых данных с регулируемыми потерями.
8. Приведите определение архиватора. Приведите примеры распространенных архиваторов.

9. Приведите формулы подсчета коэффициента сжатия, степени сжатия.
10. Приведите формулу подсчета симметричности по времени алгоритма сжатия. В каких случаях оправдано применение несимметричных по времени алгоритмов сжатия данных?
11. Поясните смысл термина «масштабирование изображений» при использовании архиваторов.
12. В рекламе на архиватор А указано, что он имеет коэффициент сжатия 20, в рекламе на архиватор В указано, что он обеспечивает степень сжатия 80 Какой из архиваторов формирует более компактный файл сжатых данных?
13. Какие алгоритмы сжатия без потерь Вам известны?
14. Сформулируйте идею сжатия данных статистическими алгоритмами.
15. Какой принцип положен в основу алгоритмов RLE?
16. Какие файлы являются наилучшими объектами для сжатия алгоритмом RLE?
17. Сформулируйте идею алгоритма кодирования по ключевым словам.
18. Для каких данных этот алгоритм наиболее эффективен?
19. Сформулируйте сущность алгоритма LZ.
20. Почему используется вариант алгоритма Лемпеля-Зива со скользящим окном?
21. Опишите процесс сжатия данных алгоритмом Лемпеля-Зива.
22. Чем отличается алгоритм арифметического кодирования от алгоритма Хаффмена?

3. ИЗУЧЕНИЕ АЛГОРИТМА СЖАТИЯ СТАНДАРТА JPEG

Цель

1. Изучить принципы реализации алгоритма сжатия стандарта JPEG, его характеристики и ограничения;

2. Совершенствовать навыки использования языков и сред программирования для реализации алгоритмов сжатия;

3. Совершенствовать навыки анализа, обобщения и систематизации полученных результатов, навыки составления и оформления отчетных материалов, навыки точного и лаконичного представления докладов на вопросы технического характера.

Учебные вопросы

1. Разработка словесного описания алгоритма, схемы алгоритма;
2. Подготовка исходного текста программы;
3. Отладка и оценка работоспособности программы сжатия.

Литература для подготовки к занятию

1. Ватолин Д. и др. Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео. - М.: ДИАЛОГ-МИФИ, 2003. - 384 с.
2. Тропченко А.Ю., Тропченко А.А. Методы сжатия изображений, аудиосигналов и видео: Учебное пособие. - СПб.: СПбГУ ИТМО, 2009. - 108 с.
3. Программная реализация дискретного косинусного преобразования алгоритма JPEG. Методические указания для проведения лабораторной работы. - Томск: Изд. ТПУ, 2000. - 16с.

Содержание отчета

1. Название работы;

2. Для каждого задания: название задания и материал в объеме, указанном в задании.

Классические алгоритмы сжатия без потерь перестали удовлетворять требованиям, предъявляемым к сжатию. Многие изображения практически не сжимались, хотя "на взгляд" обладали явной избыточностью. Это привело к созданию нового типа алгоритмов, сжимающих с потерей информации. Как правило, степень сжатия и, следовательно, степень потерь качества в них можно задавать. При этом достигается компромисс между размером и качеством изображений.

Одна из серьезных проблем машинной графики заключается в том, что до сих пор не найден адекватный критерий оценки потерь качества изобра-

жения. А теряется оно постоянно - при оцифровке, при переводе в ограниченную палитру цветов, при переводе в другую систему цветопредставления для печати и, что особенно важно, при сжатии с потерями.

JPEG (Joint Photographic Expert Group) - один из новых и достаточно мощных алгоритмов, разработанных для сжатия 24-битовых изображений. Практически он является стандартом де-факто для полноцветных изображений. Оперирует алгоритм областями 8x8, на которых яркость и цвет меняются сравнительно плавно. Вследствие этого при разложении матрицы такой, области в двойной ряд по косинусам значимыми оказываются только первые коэффициенты. Таким образом, сжатие в JPEG осуществляется за счет плавности изменения цветов в изображении.

В целом алгоритм основан на дискретном косинусном преобразовании (в дальнейшем - ДКП), применяемом к матрице изображения для получения некоторой новой матрицы коэффициентов. Для получения исходного изображения применяется обратное преобразование.

ДКП раскладывает изображение по амплитудам некоторых частот. Таким образом, при преобразовании мы получаем матрицу, в которой многие коэффициенты либо близки, либо равны нулю. Кроме того, благодаря несовершенству человеческого зрения можно аппроксимировать коэффициенты более грубо без заметной потери качества изображения.

Задание 3.1. Разработка словесного описания алгоритма, схемы алгоритма

Разработать алгоритмы, выполняющие следующие функции:

а) формирование ДКП матрицы. Вычисление осуществлять с точностью 6 знаков после запятой (точки);

б) формирование транспонированной ДКП матрицы;

в) формирование из исходной матрицы изображения (дает преподаватель) P_{DCT}-матрицу. Перед выполнением ДКП нужно из всех элементов матрицы изображения вычесть число 128; в этой части программы, предусмотреть возможность использования матрицы ДКП, получаемой по одной из возможных формул;

г) составление матрицы квантования для качества, задаваемого с клавиатуры;

д) реализация квантования, т. е. получение P^{Q}_{DCT};

е) преобразование P^{Q}_{DCT} в матрицу распакованного изображения.

В отчете представить:

1. Схема этапов работы алгоритма JPEG и краткое описание функций, выполняемых на этапах;
2. Последовательность выполнения ДКП. Формулы для подсчета элементов матриц ДКП;
3. Схемы алгоритмов преобразования на этапах JPEG;
4. Имена и типы использованных переменных.

Задание 3.2. Подготовка исходного текста программы

В соответствии с разработанным алгоритмом составить программу его реализации на одном из языков программирования. Использовать освоенные среды программирования.

Все данные (предварительно сформированные) можно поместить в один файл последовательно друг за другом. Матрицы располагаются по строкам, а векторы – в виде вектора строки. В этом случае формируется одна последовательность опреаторов для открытия файла для чтения. Возможно создать для каждой матрицы и вектора отдельные файлы. После чтения данных из файлов, все файлы должны быть закрыты.

Ввод матрицы выполняется по строкам с помощью двойного цикла.

Транспонированной матрицей AT называется матрица, у которой столбцы соответствуют строкам, а строки столбцам матрицы A, например:

$$A = \begin{vmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{vmatrix} \qquad A^T = \begin{vmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{vmatrix} .$$

Из примера видно, что, если матрица квадратная, то диагональные элементы у матрицы A и AT одни и те же, а переставляются элементы симметричные относительно главной диагонали. Эта операция реализуется операторами

```
c := aij; aij := aji; aji := c;
```

выполняемыми в двойном цикле: во внешнем по i во внутреннем по j.

Умножение матрицы A[mxn] на матрицу B[nxl] выполняется по формуле

$$C_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj} ,$$

где j = 1,2…l, i = 1,2…m, Cij – элементы матрицы C, получающейся в результате умножения. Полученная матрица C имеет размер mxl.

Для вычисления элементов матрицы C[mxl] необходимо использовать тройной цикл: по i, по j, по k.

Например при перемножении следующих матриц (m = 4, n = 5, l = 3) получаем:

$$\begin{vmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 1 \\ 2 & 3 & 4 & 5 & 6 \\ 7 & 8 & 9 & 1 & 2 \end{vmatrix} * \begin{vmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \\ 1 & 2 & 3 \\ 1 & 2 & 3 \\ 1 & 2 & 3 \end{vmatrix} = \begin{vmatrix} 15 & 30 & 45 \\ 31 & 62 & 93 \\ 20 & 40 & 81 \\ 27 & 54 & 81 \end{vmatrix}.$$

В отчете представить:

1.Исходный текст программы.

2.Комментарии к созданным блокам и процедурам.

Задание 3.3. Отладка и оценка работоспособности программы сжатия

Провести отладку программы на примере подготовленного тестового (или иного) файла для сжатия.

Выполнить тестовые прогоны для оценки параметров сжатия.

В отчете представить:

1.Отлаженную программу сжатия;

2. Распечатки матриц ДКП, полученные по формулам, соответствующих им транспонированных матриц;

3. Распечатку матрицы блока сжимаемого изображения, матрицу PDCT, полученные при использовании ДКП;

4. Данные тестовых прогонов на различных файлах (время работы, степень сжатия).

<u>Вопросы для самопроверки</u>

1. За счет чего осуществляется сжатие в алгоритме JPEG?
2. Для чего необходимо преобразование цветового пространства изображения?
3. Почему в алгоритме не применяется субдискретизация компонентов яркости?
4. Назовите виды сегментации изображений и их отличия. Какая сегментация используется в алгоритме JPEG?
5. Для чего необходимо дискретное косинусное преобразование?

6. На каком этапе сжатия происходит потеря качества изображения? Как они связаны с фактором качества?
7. Каким образом проявляются потери качества в алгоритме JPEG?
8. Какие алгоритмы кодирования использует JPEG?
9. Приведите примеры мер потери информации и опишите их недостатки.
10. В чем отличие сжатия цветных изображений от процедуры сжатия полутоновых изображений по стандарту JPEG?
11. Обнуление какого коэффициента ДКП вызовет самую сильную деградацию изображения?
12. Перечислите основные этапы работы алгоритма JPEG? Кратко сформулируйте, что делается на каждом этапе.
13. Приведите формулы перехода из цветовой системы RGB в цветовую систему YUV?
14. Приведите формулы перехода из цветовой системы YUV в систему RGB?
15. Объясните цель субдискретизации. Какие особенности зрения человека используются?
16. Для каких изображений субдискретизация не проводится?
17. Цель и порядок выполнения дискретного косинусного преобразования (ДКП)?
18. В каких стандартных алгоритмах сжатия кроме JPEG используется ДКП?
19. На каких этапах работы JPEG происходят невосстанавливаемые потери информации?
20. Нужно ли при сжатии запоминать значение «качество»? Если нужно, то – для чего?
21. Обязательно ли JPEG предполагает сжатие с потерями?
22. JPEG – открытый или закрытый стандарт? Может ли разработчик программного продукта внести какие-то свои изменения?
23. Каким образом подсчитывается коэффициент сжатия?
24. Каким образом подсчитывается степень сжатия?

4. ИЗУЧЕНИЕ АРХИВАТОРОВ

Цель

1. Изучить принципы реализации архиваторов, их особенности и параметры эффективности сжатия;

2. Совершенствовать навыки анализа, обобщения и систематизации полученных результатов, навыки составления и оформления отчетных материалов, навыки точного и лаконичного представления докладов на вопросы технического характера.

Учебные вопросы

1. Работа с архиватором;
2. Исследовать сжимаемость файлов различных типов;
3. Исследовать влияние метода сжатия на коэффициент сжатия и время сжатия;
4. Исследовать влияние размера словаря на коэффициент сжатия.

Литература для подготовки к занятию

1. Ватолин Д. и др. Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео. - М.: ДИАЛОГ-МИФИ, 2003. - 384с.
2. Тропченко А.Ю., Тропченко А.А. Методы сжатия изображений, аудиосигналов и видео: Учебное пособие. - СПб.: СПб ГУИТМО, 2009. - 108с.

Актуальность занятия

1. Многие виды информации в настоящее время просто не могут сохраняться и обрабатываться в несжатом виде (велико время, например, для видеосигналов);
2. Архиваторы – общедоступный универсальный инструмент сохранения всех необходимых данных для обработки информации. Используется массово. Информация в настоящее время сохраняется практически всегда в заархивированном виде (инсталляции).
3. Теоретические положения сжатия применяются для скрытия признаков обмена данными.

Содержание отчета

1. Название работы;

2. Для каждого задания: название задания и материал в объеме, указанном в задании.

Задание 4.1. Работа с архиватором

Изучить и выполнить основные операции с архиватором.
Индивидуальные задания:

В папке ***"Мои документы/ФАМ"*** создать папку"**АрхивФАМ**"
(файлы, необходимые для архивирования подобрать по указанию преподавателя).

В отчете представить:
1. Перечень типовых операций с архиватором;
2. Созданный архив.

WinRAR—это мощное средство создания архивов и управления ими.

Возможности WinRAR:

- полная поддержка архивов RAR и ZIP;
- оригинальный высокоэффективный алгоритм сжатия данных;
- специальный алгоритм мультимедиа сжатия;
- поддержка технологии перетащить-и-оставить (drag&drop);
- интерфейс командной строки;
- управление архивами других форматов;
- метод непрерывного (solid) архивирования, при использовании которого может быть достигнута на 10–50% более высокая степень сжатия, чем дают обычные методы;
- поддержка многотомных архивов;
- создание самораспаковывающихся (SFX) обычных и многотомных архивов с помощью стандартного или дополнительных модулей SFX;
- восстановление физически поврежденных архивов;
- поддержка кодировки Unicode в именах файлов;
- другие дополнительные функции, например, шифрование, добавление архивных комментариев и пр.

Интерфейс и основные инструменты

Доступ к WinRar можно получить двумя путями:
- используя меню "Пуск" (рис.4.1,а);

- выполнив щелчок ПКМ по файлу или папке, которую необходимо архивировать и из выпавшего списка выбрать пункт "добавить в архив" (рис.4.1,б).

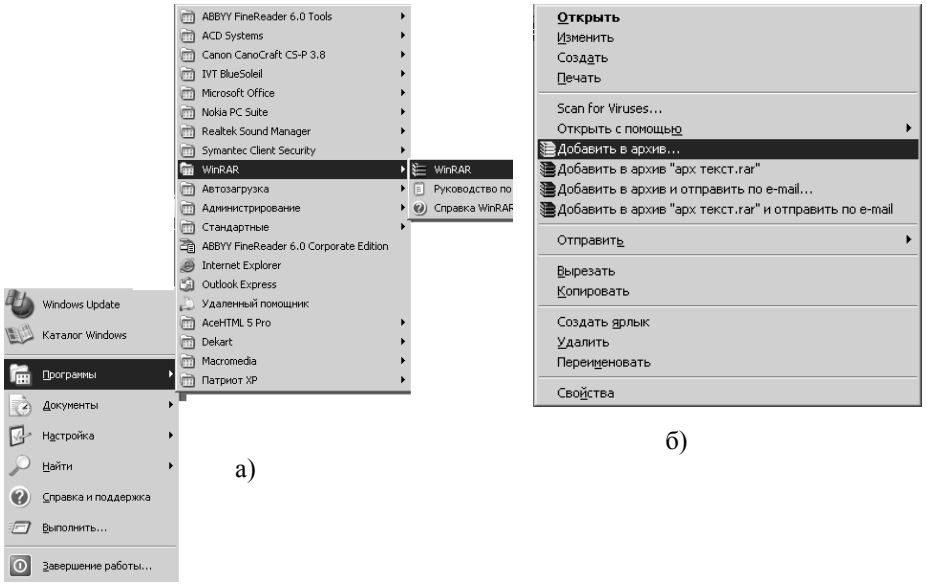


Рис.4.1. Способы доступа к архиватору: а) через меню ПУСК; б) из контекстного меню

Если используется первый способ (рис.4.1,а), то открывается стандартное окно архиватора (рис.4.2).

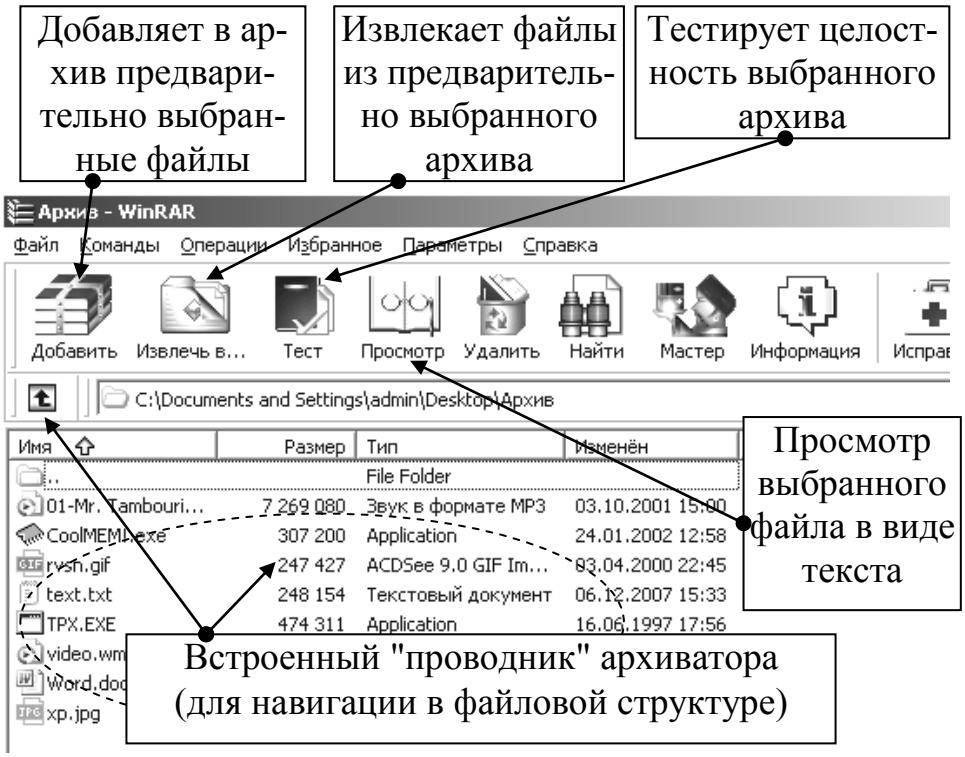


Рис.4.2. Стандартное окно архиватора

Архивация файлов (группы файлов)

Архивировать (упаковывать в архив) можно отдельные файлы, группы файлов, папки с файлами.

Перед началом архивации необходимо определить параметры будущего архива:

- состав архива (какие файлы будут входить в архив);
- вид архива: обычный или самораспаковывающийся;
- размер словаря, необходимого для архива;
- конкретное место (диск, папка), в котором будет создан архив.

Общий порядок архивирования файла

Для создания архива необходимо: открыть архиватор, с помощью встроенного проводника найти файл, который следует упаковать в архив, выбрать параметры архивации, указать место, в котором будет создан архив и завершить операцию архивации.

Пример 1.

Создать архив на рабочем столе. Метод сжатия – максимальный. В состав архива включить файл kalendar_2007.doc, находящийся по адресу D:\doc\KAG.

Решение.

1. Открыть WinRar и выполнить последующие шаги как показано на рис.4.3.

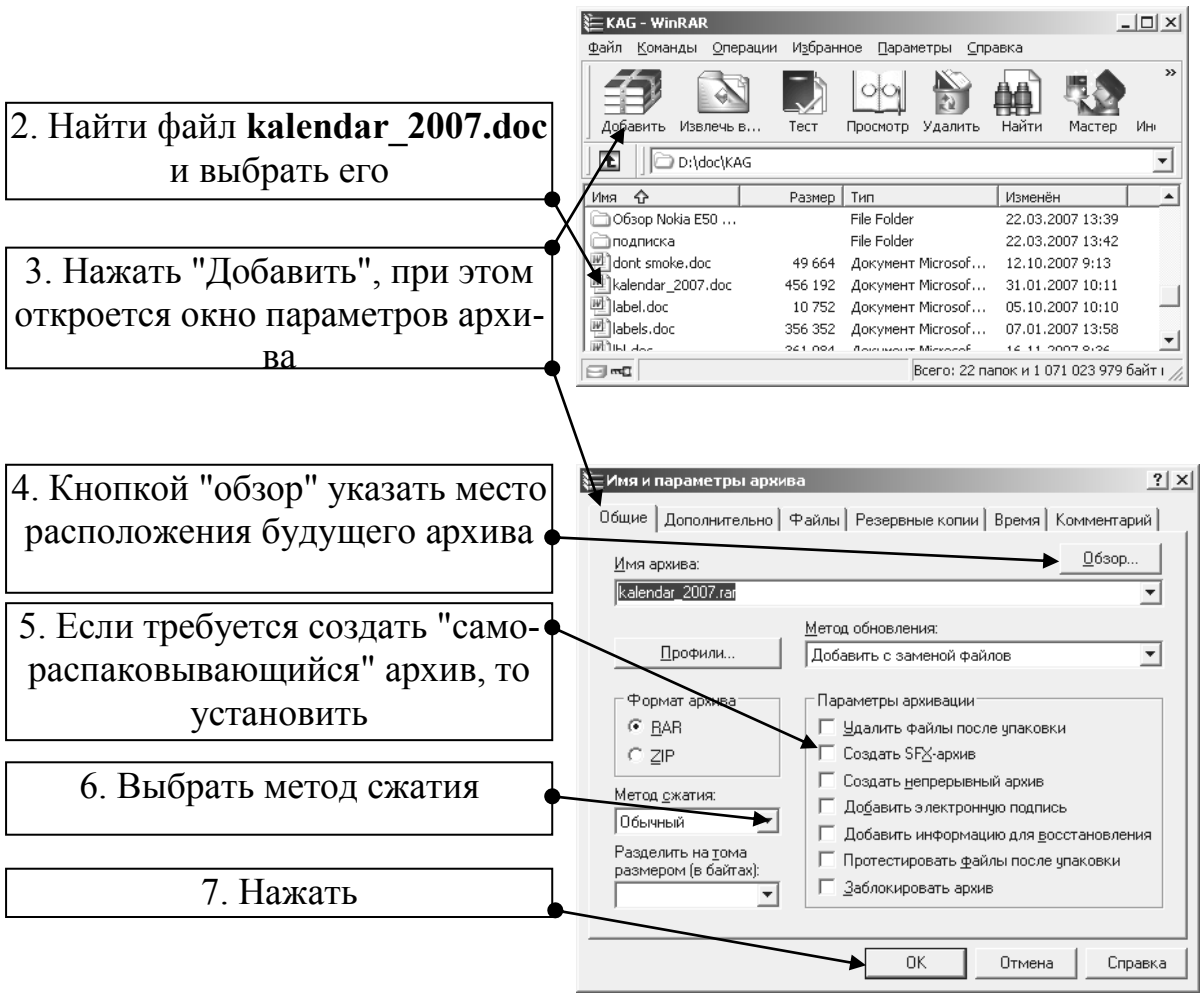


Рис.4.3. Этапы создания архива

В результате этих действий на рабочем столе появится архив **kalendar_2007.rar** (рис.4.4). В случае создания самораспаковывающегося архива создаётся файл **kalendar_2007.exe** (рис.4.5).

Рис.4.4. Архив RAR

Рис.4.5. Самораспаковывающийся архив

Если требуется создать архив с **определённым размером словаря**, то перед тем, как нажать кнопку **Ок** (см.рис.4.3) необходимо в окне "Имя и параметры архива" выбрать вкладку" **Дополнительно**" и в открывшемся окне нажать кнопку "**Параметры сжатия**" и установить размер словаря (рис.4.6).

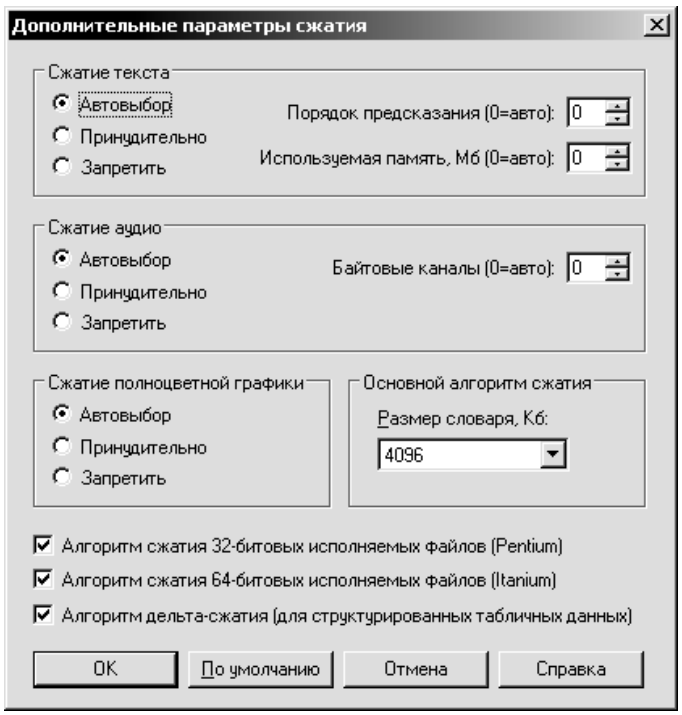


Рис.4.6. Настройка параметров сжатия

Общий порядок разархивирования (распаковки) архива

Перед извлечением файлов из архива необходимо выяснить, в каком месте на диске должен оказаться распакованный файл (группа файлов).

Мнение о том, что, якобы, сначала можно распаковать, а затем переместить в другое место – ошибочно, так как, во-первых – может тривиально не хватить места на диске для хранения распакованного архива, а во-вторых – к папке, в которой находится архив, может быть организован доступ "только чтение", что не позволит сохранить распакованные файлы.

Пример 2.

На диске **D** имеется архив **kalendar_2007.rar.** Необходимо распаковать архив в папку **buffer**, которая находится на диске **D**.

Решение.

1. Открыть архиватор WinRar.
2. С помощью проводника архиватора найти архив **kalendar_2007.rar**.
3. В окне архиватора выполнить двойной щелчок ЛКМ по найденному архиву, при этом архив раскроется и пользователь **увидит какие файлы входят в состав архива**. Далее действуйте по этапам рис.4.7.

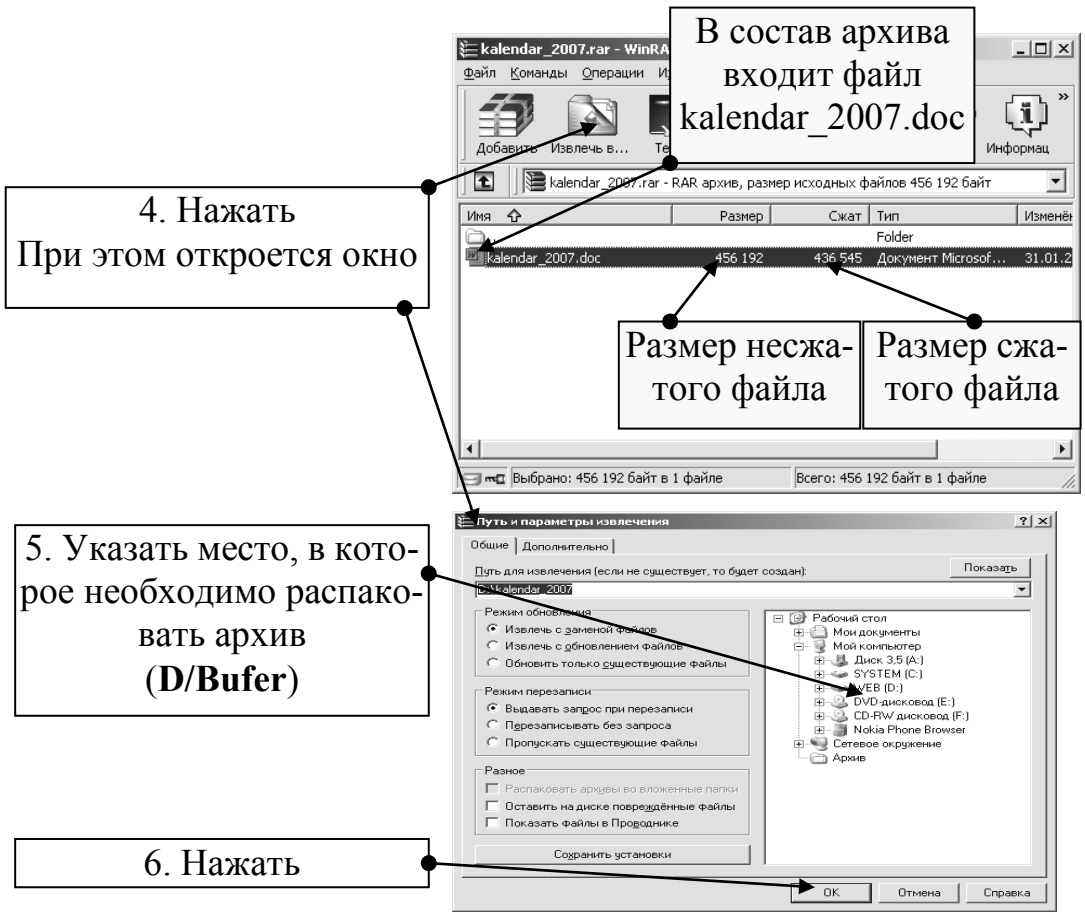


Рис.4.7. Завершающие этапы распаковки

В результате выполнения п.п.1–6 в папке **buffer** появится файл **kalendar_2007.doc**.

Особенности распаковки самораспаковывающегося архива

Самораспаковывающиеся архивы предназначены для распаковки в компьютерах, на которых не установлена программа- архиватор, либо версия архиватора более ранняя и не способна распаковать архив, созданный в более поздней версии архиватора. Общий недостаток этого вида архивов - несколько больший размер архивного файла за счёт того, что в нём помимо заархивированных файлов содержится фрагмент программы- архиватора, который и осуществляет распаковку архива.

Для распаковки такого архива необходимо выполнить двойной щелчок по самораспаковывающемуся архиву (он имеет расширение **.exe**, что воспринимается операционной системой как приложение), в открывшемся окне указать место, в которое необходимо распаковать архив, и нажать кнопку "**Извлечь**".

Изменение состава архива

Состав уже созданного архива может быть изменён. Под изменением будем понимать добавление новых файлов, или удаление существующих файлов из архива.

Общий порядок изменения состава архива

1. Открыть архиватор WinRar.
2. Найти архив, состав которого требуется изменить.
3. Выполнить двойной щелчок ЛКМ по значку найденного архива

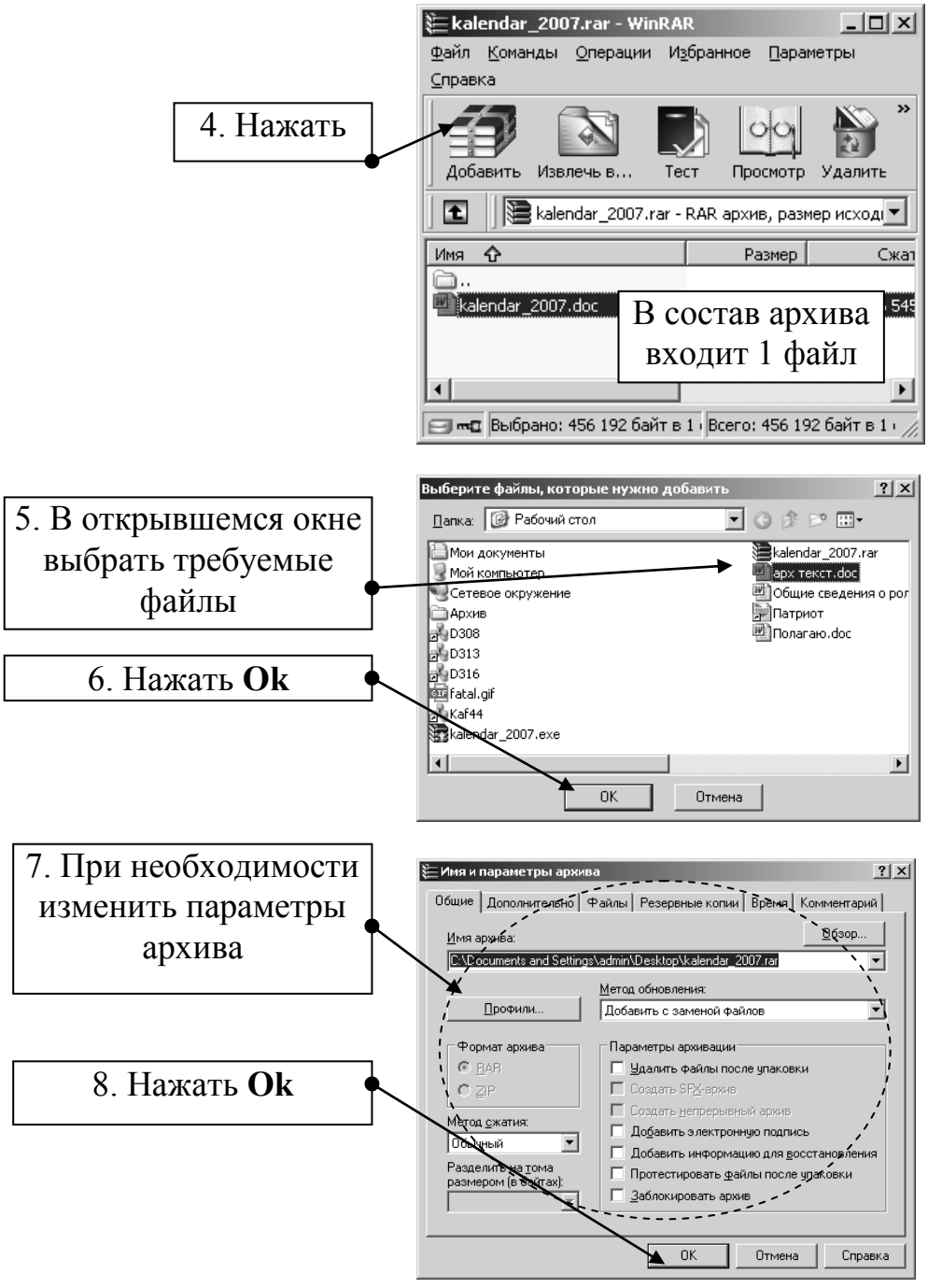


Рис.4.8. Изменение состава архива

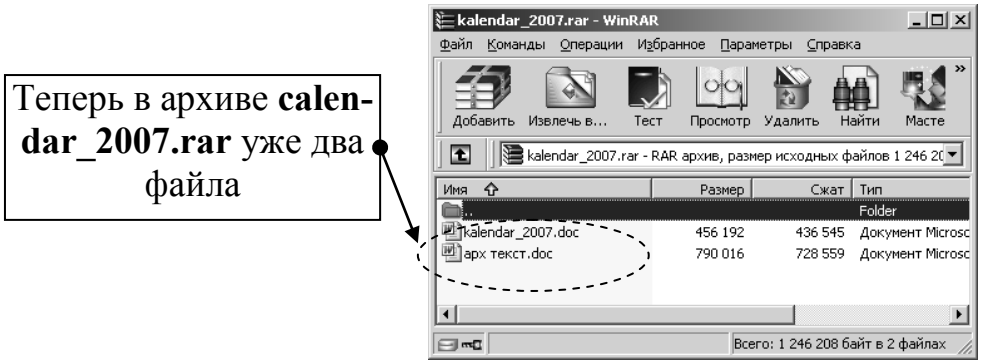


Рис.4.8. Продолжение

Если необходимо **удалить** некоторые файлы из архива, то необходимо:
В окне архиватора открыть архив и просмотреть, какие файлы входят в его состав.
Выделить ЛКМ удаляемый файл и нажать кнопку "**Удалить**" на панели инструментов архиватора, при этом из состава архива удаляется выбранный файл.

Задание

Создать самораспаковывающийся архив

1. Создать самораспаковывающийся архив из файла **Word.doc**. (Архив должен быть создан в папке **АрхивФИО**).
2. Извлечь файлы из созданного архива (файлы должны сразу оказаться на Рабочем столе)

Изменить состав архива

1. В созданный при выполнении предыдущих задача рхив **music.rar** добавьте любые 1-2 файла.
2. В созданный самораспаковывающийся архив добавьте любые 1-2 файла.
3. Просмотрите содержимое получившегося архива и извлеките из него только один из входящих в его состав файлов.
4. Удалите из состава этого архива любой файл.

Задание 4.2. Исследовать сжимаемость файлов различных типов

1. Создать непрерывные архивы файлов: **music.mp3**, **video.wmv**, **text.txt**, **Word.doc**, **XP.gif** (архивы должны быть созданы сразу в папке **АрхивФАМ**). Метод сжатия – **максимальный**, размер словаря – везде одинаковый.

2. Записать в конспект размеры: исходного файла и сжатого файла во всех созданных архивах.

3. Результаты оформить в видета блицы Excel с использованием формулы расчёта коэффициента сжатия:

Имя файла	Исходный размер (байт)	Размер архива (байт)	Коэфф. сжатия

Коэфф.сжатия = Исходный размер/Размер архива

В отчете представить:
1. Таблицу эксперимента в формате Excel;
2. Графическое представление результатов в Excel;
3. Выводы о причине таких результатов.

Задание 4.3. Исследовать влияние метода сжатия на коэффициент сжатия и время сжатия

1. Создать непрерывные архивы файла music.mp3, изменяя методы сжатия: **скоростной**, **быстрый**, **обычный**, **хороший**, **максимальный**. Время сжатия показывается в процессе создания архива.

2. Записать: исходный размер файла, метод сжатия, время сжатия и размер архива.

3. Результаты оформить в виде таблицы Excel с использованием формулы расчёта коэффициента сжатия:

Метод сжатия	Исходный размер(байт)	Размер архива(байт)	Коэфф. сжатия	Время сжатия(с)

В отчете представить:
1. Таблицу экспериметна в формате Excel;
2. Графическое представление результатов в Excel;

3. Выводы о причине таких результатов.

Задание 4.4. Исследовать влияние размера словаря на коэффициент сжатия

1. Создать непрерывные архивы файла music.mp3, изменяя размер словаря: **64**, **128**, **256**, **512**, **1024**, **2048**, **4096**.

2. Записать: исходный размер файла, размер словаря, размер архива.

3. Результаты оформить в виде таблицы Excel с использованием формулы расчёта коэффициента сжатия:

Размер словаря(кБ)	Исходный размер(байт)	Размер архива(байт)	Коэфф.сжатия

В отчете представить:
1. Таблицу эксперимента в формате Excel;
2. Графическое представление результатов в Excel;
3. Выводы о причине таких результатов.

Вопросы для самопроверки

1. Для чего создаются архивы?
2. Что такое самораспаковывающийся архив?
3. Перечислите различные способы запуска архиватора 7Zip.
4. Какое расширение имеет архив, созданный стандартным архиватором Windows?
5. Перечислите, какие архиваторы Вам известны.
6. Какие файла поддаются более эффективному сжатию? Почему?
7. Какие параметры архиватора настраиваются?
8. Поясните отличие операции создания архива и добавления в архив.
9. Какие расширения имеют архив и самораспаковывающийся архив?
10. Возможна ли пересылка архива по электронной почте?
11. Как защитить данные архива?
12. Что такое словарь архива и на что может повлиять его размер?
13. Практически все архиваторы обеспечивают возможность просмотра файлов, содержащихся в архивах. Почему эти файлы нельзя редактировать?

14. Какая информация обязательно содержится в оглавлении архивного файла?