Лекция 9

23.0 **Сжатие данных**

Хранение, передача информации участникам информационного процесса обходятся недёшево. В связи с этим, возникает необходимость сжимать данные перед тем, как размещать их в архивах или передать по каналам связи. Соответственно, существует и обратная необходимость восстановления данных из предварительно уплотнённых архивов.

Большинство данных содержат **избыточность**. Избыточность информации улучшает её восприятие и обработку. При хранении информации или её передачи избыточность можно уменьшить с помощью сжатия данных.

Степень избыточности зависит от типа данных и от принятой системы кодирования.

Существующие типы данных:

1. ***Видеоданные данные*** обладают самой большой избыточностью;
2. ***Графические данные –*** средней избыточностью;
3. ***Текстовые данные*** – низкой избыточностью.

Система кодирования текстовой информации средствами русского языка (русской азбуки) даёт избыточность на 20 – 30% больше, чем кодирование адекватной информации средствами английского языка.

Сжатие данных – это процесс преобразования информации, при котором уменьшается избыточность в её представлении и соответственно требуется меньший объём памяти для хранения.

Если методы сжатия информации применяют к готовым документам, то термин *сжатие данных* заменяют термином *архивация данных*, а программные средства, выполняющие эти операции, называют *архиваторами*.

Уплотнение в один архивный файл группы файлов

- существенно упрощает их перенос с одного компьютера на другой,
- сокращает время копирования файлов на диски,
- позволяет защитить информацию от несанкционированного доступа,
- способствует защите от заражения компьютерными вирусами.

В зависимости от объекта расположения данных, подвергаемых сжатию различают:

- уплотнение (архивацию) файлов;
- уплотнение (архивацию) папок;
- уплотнение дисков.

Уплотнение файлов применяют для уменьшения их размеров при подготовке к передаче по каналам электронной сети или транспортировке на диске.

Архивный файл содержит в себе один или несколько файлов в сжатом виде и служебную информацию об именах файлов, дате и времени их создания или модификации, размерах и т.п.

Степень сжатия файлов характеризуется отношением:

$$S_C = \frac{V_O}{V_C},$$

где V_c – объём сжатого файла
V_o – объём исходного файла

Значения Sc больше 1 обозначают сжатие, а значения меньше 1 – расширение.

$$S_C = \frac{1500\ \text{кБт}}{300\ \text{кБт}} = 5 \quad \text{раз}$$

Коэффициент сжатия – это величина, обратная степени сжатия.

$$K_C = \frac{V_C}{V_O} * 100\%, \qquad K_C = \frac{300\ \text{кБт}}{1500\ \text{кБт}} * 100\% = 20\%$$

Например, *значение* 0,6 означает, что данные занимают 60% от первоначального объема. Значения больше 1 (100%) означают, что выходной *поток* больше входного (отрицательное сжатие, или расширение)

Степень сжатия объектов зависит от следующих факторов:

- метода сжатия;
- используемой программы;
- типа исходного файла.

Хорошо сжимаются видео-, графические, некоторые текстовые файлы, ($К_с$=5-40%).
Меньше сжимаются файлы исполняемых программ и загрузочных модулей (60-90 %).
Почти не сжимаются архивные файлы.

Уплотнение папок используют как средство архивации данных перед длительным хранением, в частности, при резервном копировании.

Уплотнение дисков служит целям повышения эффективности использования их рабочего пространства и применяется к дискам, имеющим недостаточную ёмкость.

23.1 ***Обратимость сжатия***

Существуют 3 способа уменьшения избыточности данных:

1. Изменение содержания данных;
2. Изменение структуры данных;
3. Изменение содержания и структуры данных.

1. Если при сжатии данных происходит изменение их содержания, метод сжатия **не обратим** и при восстановлении данных, полного восстановления исходной последовательности, не происходит.

Такие методы называют ***методами сжатия с регулируемой потерей информации.*** Они применяются только для тех типов данных, для которых формальная утрата части содержания не приводит к значительному снижению потребительских свойств.

Эти методы обеспечивают гораздо более высокую степень сжатия, чем обратимые методы, но их нельзя применять к текстовым документам, базам данных, программному коду.

Их применяют к музыкальным записям, видеозаписям и рисункам.

Форматы сжатия с потерей информации:

- ***.JPG** для ***графических*** данных;
- ***.MPG** для ***видеоданных***;
- ***.MP3** для ***звуковых*** данных.

2. Если при сжатии данных происходит изменение их структуры, то метод сжатия **обратим** и можно восстановить исходный массив путём применения обратного метода.

Обратимые методы применяются для сжатия любых типов данных.

Форматы сжатия без потери информации:

- ***. GIF**, ***.TIF**, ***.PCX** и другие для ***графических*** данных;
- ***.AVI** для ***видеоданных***;
- ***.ZIP**, ***.ARJ**, ***.RAR**, ***.LH**, ***.LZH**, ***.CAB** и другие для ***любых*** типов данных.

При сжатии данных следует иметь ввиду, что:

- Для любой последовательности данных существует ***теоретический предел сжатия***, который не может быть превышен без потери части информации;
- Для любого алгоритма сжатия можно указать такую последовательность данных, для которой он обеспечит лучшую степень сжатия, чем другие методы;
- Для любого алгоритма сжатия можно указать такую последовательность данных, для которой данный алгоритм вообще не позволит получить сжатие.

23.2 *Алгоритмы обратимых методов*

Существует много обратимых методов сжатия данных, наивысшую эффективность они демонстрируют для данных разных типов и разных объёмов.

В основе методов сжатия лежат следующие алгоритмы:

RLE (Run-Length Encoding) – для ***графических*** данных;

KWE (Keyword Encoding) – для ***текстовых*** данных;

Алгоритм **Хаффмана** – для ***любых*** данных.

Алгоритм RLE

В основу положен принцип выявления повторяющихся последовательностей данных и замены их простой структурой, в которой указывается ***код данных*** и ***коэффициент повтора.***

Например, для последовательности:

0; 0; 0; 125;125; 0; 248; 248; 248; 248 (всего 10 байтов)

1 2 3 4 5 6 7 8 9 10

Выявляются: значения и коэффициент повтора.

0	3
125	2
0	1
248	4

Новый вектор данных имеет вид:

0; 3; 125; 2; 0; 1; 248; 4 (всего 8 байтов)

Коэффициент сжатия Кс=8/10*100%=80% или степень сжатия 10/8=1,25 раз.

Алгоритм отличается простотой, высокой скоростью работы, но обеспечивает недостаточное сжатие.

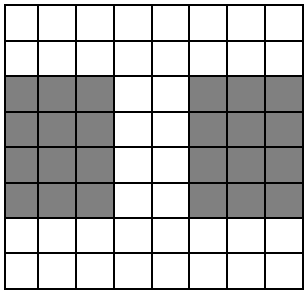
Наилучшими объектами для данного алгоритма являются **графические файлы**, в которых большие одноцветные участки изображения кодируются длинными последовательностями одинаковых байтов.

Этот метод может дать заметный **выигрыш на некоторых типах файлов баз данных**. Имеющих таблицы с фиксированной длиной полей.

Для ***текстовых данных методы RLE неэффективны***.

В связи с этим большая эффективность алгоритма RLE достигается при сжатии графических данных (в особенности для однотонных изображений).

Пример сжатия серно-белого изображения



1 клетка 8 бит (Будем кодировать 0 – черный /или другой/ цвет – 8 бит (2^8 =256 цветов), 1 – белый цвет – 8 бит), весь рисунок занимает памяти 8*64=512 бит.

	Применяя сжатие	Без сжатия/Со сжатием
Для фигуры первая строка	→ 1(б),8(раз) = 2 байта	64 бита 16 бит
вторая строка	→ 1(б),8(раз) = 2 байта	64 бита 16 бит
третья строка	→ 0(ч),3(р)+1,2+0,3 = 6 байт	64 бита 48 бит
четвертая строка	→ 0,3+1,2+0,3 = 6 байт	64 бита 48 бит
пятая строка	→ 0,3+1,2+0,3 = 6 байт	64 бита 48 бит
шестая строка	→ 0,3+1,2+1,3 = 6 байт	64 бита 48 бит
седьмая строка	→ 1,8 = 2 байта	64 бита 16 бит
восьмая строка	→ 1,8 =2 байта	64 бита 16 бит

Объем рисунка до сжатия 8 строк*64бит=512 бит

Объем рисунка после сжатия 4*2(байта)+4*6(байт)=8+24=32 байт*8=256 бит

Кс=256*100/512 = 50%

№ строки	Vo исходного изображения, бит	Vсжатого 8-разрядного изображения Цвет/коэффициент повтора
1	8*8=64	1,8=2 байт=16 бит
2	8*8=64	1,8=2 байт=16 бит
3	8*8=64	0,3+1,2+1,3 =6 байт=48 бит
4	8*8=64	0,3+1,2+1,3 =6 байт=48 бит
5	8*8=64	0,3+1,2+1,3 =6 байт=48 бит
6	8*8=64	0,3+1,2+1,3 =6 байт=48 бит
7	8*8=64	1,8=2 байт=16 бит
8	8*8=64	1,8=2 байт=16 бит
Итого	8*64=512 бит	4*16+4*48=64+192=256 бит

Кс=256*100/512 = 50%

Алгоритм KWE

В основу положен принцип кодирования лексических единиц исходного документа группами байтов фиксированной длины.

Лексическая единица – последовательность символов, справа и слева ограниченная пробелами или символами конца абзаца.

Лексической единицей может быть ***слово***.

Результат кодирования сводится в таблицу, которая прикладывается к исходному коду и представляет собой словарь.

Для англоязычных текстов принято использовать двухбайтовую кодировку слов.

В русском языке много длинных слов, большое количество приставок, суффиксов и окончаний, поэтому не всегда удаётся ограничиться 2-х байтовыми словами. Образующиеся при этом пары байтов называют токенами.

Эффективность метода зависит от длины документа.

Из-за необходимости прикладывать к архиву словарь слов, длина кратких документов не только не уменьшается, но даже возрастает.

Алгоритм наиболее **эффективен для англоязычных текстовых документов и файлов баз данных**.

Пример сжатия текстовой фразы: «Мы изучаем информатику»

Лексическая единица	2-х байтовый код
Мы	0000000011111111
изучаем	0101010101010101
информатику	0011001100110011
пробел	0000000000000000

Объем фразы до сжатия 22 символа*8бит=176 бит

Объем фразы после сжатия 5 лексических единиц*16=80 бит (без словаря)

0000000011111111**0000000000000000**0101010101010101**0000000000000000**0011001100110011

$$Кс=80*100/176 = 45,45\%$$

Алгоритм Хаффмана

В основе алгоритма лежит кодирование не байтами, а битовыми группами.

Этот *алгоритм кодирования информации* был предложен Д.А. Хаффманом в 1952 году. **Хаффмановское кодирование (сжатие)** – это широко используемый метод сжатия, присваивающий *символам алфавита* коды переменной длины, основываясь на вероятностях появления этих символов.

Алгоритм Хаффмана основывается на том, что символы в тексте, как правило, встречаются с различной частотой.

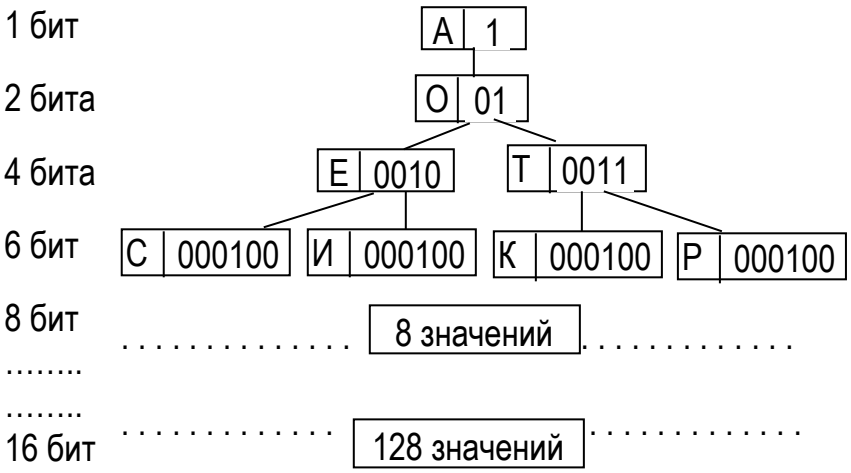
При обычном кодировании мы каждый символ записываем в фиксированном количестве бит, например, каждый символ в 1 байте, или в 2 байтах.

Однако, т. к. некоторые символы встречаются чаще, а некоторые реже − можно записать часто встречающиеся символы в небольшом количестве бит, а для редко встречающихся символов использовать более длинные коды. Тогда суммарная длина закодированного текста может стать меньше.

Принцип кодирования:

- Перед началом кодирования производится частотный анализ кода документа с целью выявления частоты повтора каждого из встречающихся символов.
- Чем чаще встречается тот или иной символ, тем меньшим количеством битов он кодируется.
- Образующаяся в результате кодирования иерархическая структура, прикладывается к сжатому документу в качестве таблицы соответствия.

Соответственно, чем реже встречается символ, тем длиннее его кодовая битовая последовательность. Например,



Используя 16 бит, можно закодировать до 256 различных символов. К сжатому архиву необходимо прикладывать таблицу соответствия, поэтому **для файлов малых размеров алгоритм Хаффмана малоэффективен.**

Эффективность алгоритма зависит и от заданной предельной длины кода (размера словаря). Наиболее эффективными оказываются архивы **с размером словаря от 512 до 1024 единиц**.

Все рассмотренные алгоритмы в «чистом виде» на практике не применяются, т.к. эффект каждого из них сильно зависит от начальных условий.

Идея алгоритма Хаффмана состоит в следующем: зная вероятности вхождения символов в исходный текст, можно описать процедуру построения кодов переменной длины, состоящих из целого количества битов. Символам с большей вероятностью присваиваются более короткие коды. Таким образом, в этом методе при сжатии данных каждому символу присваивается оптимальный *префиксный код*, основанный на вероятности его появления в тексте.

Префиксный код – это код, в котором никакое *кодовое слово* не является префиксом любого другого кодового слова. Эти коды имеют переменную длину.

Оптимальный префиксный код – это *префиксный код*, имеющий минимальную среднюю длину.

Реализацию Алгоритма Хаффмана можно разделить на два этапа:

1. Определение вероятности появления символов в исходном тексте.

   Первоначально необходимо прочитать исходный текст полностью и подсчитать вероятности появления символов в нем (иногда подсчитывают, сколько раз встречается каждый символ). Если при этом учитываются все 256 символов, то не будет разницы в сжатии текстового или файла иного формата.

2. Нахождение оптимального префиксного кода.

Коды Хаффмана имеют уникальный *префикс*, что и позволяет однозначно их декодировать, несмотря на их переменную длину.

Алгоритм построения дерева Хаффмана.

Шаг 1. Символы входного алфавита образуют *список* свободных узлов.

Каждый *узел* имеет *вес*, который может быть равен либо вероятности, либо по числу появления символа в сжимаемом тексте.

Шаг 2. Выбираются два свободных узла дерева с наименьшими весами.

Шаг 3. Создается их родитель с весом, равным их суммарному весу.

Шаг 4. Родитель добавляется в *список* свободных узлов, а двое его детей удаляются из этого списка.

Шаг 5. Одной *дуге*, выходящей из родителя, ставится в соответствие *бит* 1, другой – *бит* 0.

Шаг 6. Повторяются шаги, начиная со второго, до тех пор, пока в списке свободных узлов не останется только один свободный узел. Он и будет считаться *корнем дерева*.

Существует два подхода к построению кодового дерева: от корня к листьям и от листьев к корню.

Пример построения кодового дерева.

Пусть задана исходная последовательность символов:

```
aabbbbbbbbccccdeeeee.
```

```
 2       8       4  1   5          Количество повторений символов
```

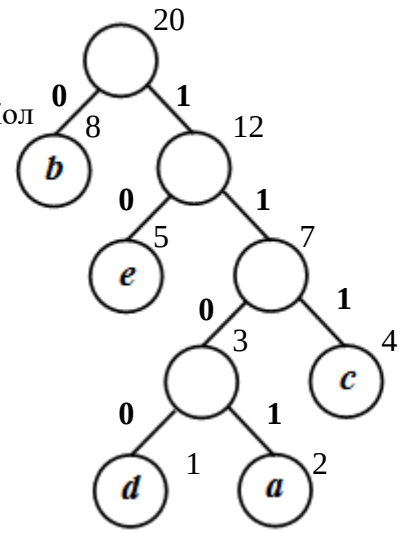
Ее исходный объем равен 20 *байт* (20*8=160 *бит*).

p=Кол/Объем 2/20=0,1… 8/20=0,4… 4/20=0,2… 1*20=0,05…5/20=0,25

Вероятности появления символов

Символ	*Вероятность*	Кол
a	0,1	2
b	0,4	8
c	0,2	4
d	0,05	1
e	0,25	5

Кодовое дерево



Оптимальные префиксные коды

Символ	*Код*
a	1101
b	0
c	111
d	1100
e	10

Создание оптимальных префиксных кодов

В соответствии с приведенными на рисунке данными (*таблица* вероятности появления символов, кодовое *дерево* и *таблица* оптимальных *префиксных кодов*) закодированная исходная последовательность символов будет выглядеть следующим образом:

1011110**00000000**1111111111111**1100**1010101010.

a a bb bb bb bb c c c c d e e e e e

Следовательно, ее объем будет равен 42 бита. Степень сжатия приближенно равен 20*8/42=160/42=3,8 раз. Или 42/160*100%=26,2 %

Классический *алгоритм* Хаффмана имеет один существенный недостаток. Для восстановления содержимого сжатого текста **при *декодировании* необходимо знать таблицу частот**, которую использовали при кодировании. Следовательно,

1. *Длина* сжатого текста увеличивается на длину таблицы частот, которая должна посылаться впереди данных, что может свести на нет все усилия по сжатию данных, если у них малый объем.
2. Необходимость наличия полной частотной статистики перед началом собственно кодирования требует двух проходов по тексту: одного для построения модели текста (таблицы частот и дерева Хаффмана), другого для собственно кодирования.

Свойства алгоритмов сжатия

Алгоритм	Выходная структура	Сфера применения	Примечание
RLE (Run-Length Encoding	Список (вектор данных)	Графические данные	Эффективность алгоритма не зависит от объема данных
KWE (Keyword Encoding)	Таблица данных (словарь)	Текстовые данные	Эффективен для массивов большого объема
Алгоритм Хаффмана	Иерархическая структура (дерево кодировки)	Любые данные	Эффективен для массивов большого объема

Синтетические алгоритмы

На практике используются более сложные алгоритмы, основанные на комбинации нескольких теоретических методов.

Общим принципом в работе «синтетических» алгоритмов является предварительный просмотр и анализ исходных данных для индивидуальной настройки алгоритма на особенности обрабатываемого материала.

23.3 *Алгоритмы необратимых методов*

Сжатие без потерь, основанное на принципе сокращения статистической избыточности, то есть более рационального размещения данных в файле, позволяет уменьшить его размер. Например, для аудиоданных даже при использовании достаточно сложных процедур обработки устранение статистической избыточности позволяет в конечном итоге уменьшить требуемую пропускную способность канала связи лишь на 20% по сравнению с ее исходной величиной.

Идея, лежащая в основе всех **алгоритмов сжатия с потерями**, довольно проста:

1 этап – удаление несущественной информации,

2 этап – к оставшимся данным применить наиболее подходящий алгоритм сжатия без потерь.

Основные сложности заключаются в выделении **несущественной** информации. Подходы здесь существенно различаются в зависимости от типа сжимаемых данных.

- Для звука чаще всего удаляют частоты, которые человек просто не способен воспринять, уменьшают частоту дискретизации, а также некоторые алгоритмы удаляют тихие звуки, следующие сразу за громкими тонами,
- Для видеоданных кодируют только движущиеся объекты, а незначительные изменения на неподвижных объектах просто отбрасывают.

Сжатие с потерями разработано на основании алгоритмов сокращения психоакустической и визуальной избыточности, то есть **удаления из файла той части звуковой и видео информации, которая незначительно влияет на восприятие звука и изображения человеком**. Этими методами можно обеспечить сжатие цифровых данных в 10 – 12 раз без существенных потерь в качестве звучания.

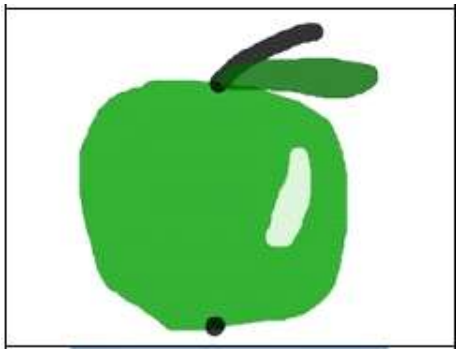
Сжатие аудиоданных с использованием психоакустического восприятия подразумевает способ преобразования потока данных, при котором кодированию подвергается только та часть звуковой информации, которую способно воспринять ухо человека, остальные же составляющие исходного сигнала можно отбросить.

В настоящее время существуют форматы сжатых звуковых файлов учитывающих психоакустическую избыточность, например MPEG, Dolby AC-3..

Сжатие видео изображения методом квантования и дискретизации связано с понятием визуальной избыточности. Значительная часть информации на изображении не может быть воспринята человеком: например, **человек способен замечать незначительные перепады яркости, но гораздо менее чувствителен к цветности**. Также, начиная с определённого момента, повышение точности дискретизации не влияет на визуальное восприятие изображения. Некоторая часть информации (визуальная избыточность) может быть удалена без ухудшения визуального качества.

.

Исходная частота дискретизации



Пониженная частота дискретизации

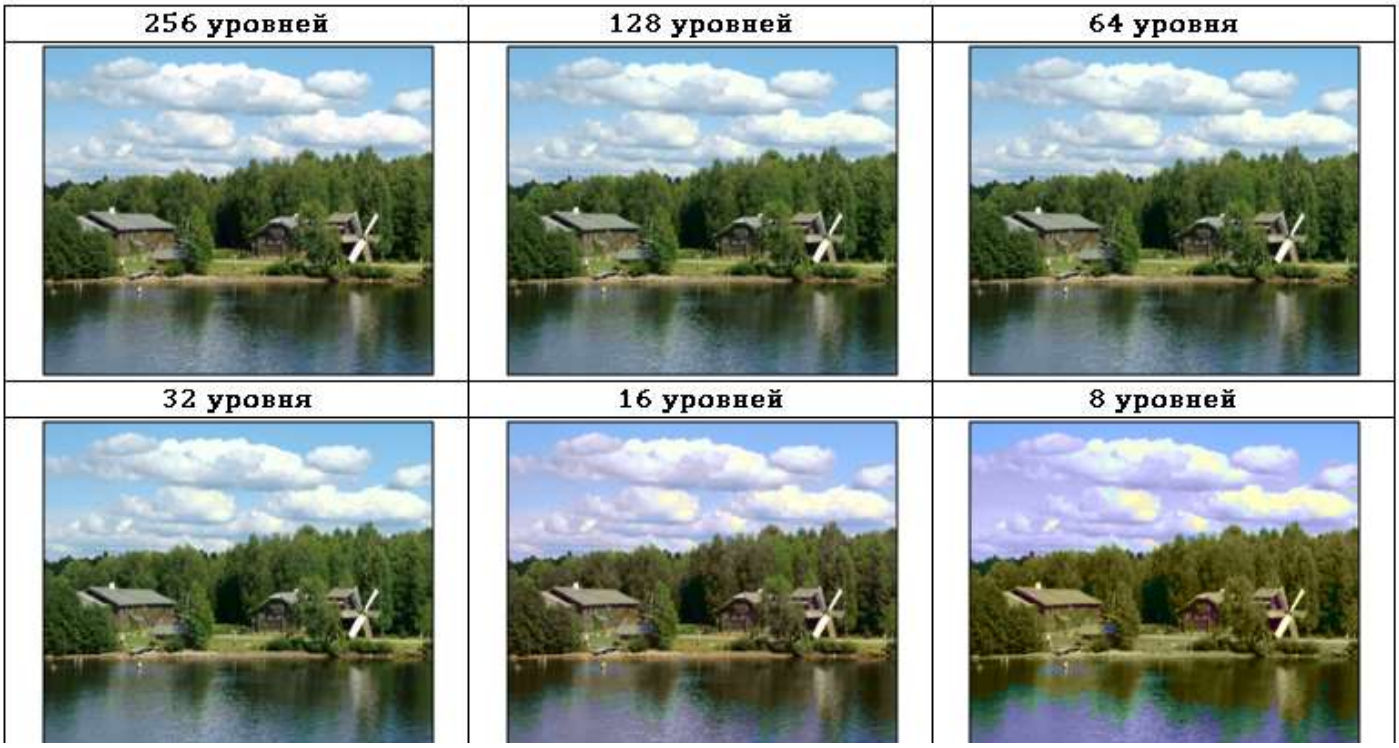
Сжатие в 100 раз, но с худшим качеством.

Для удаления избыточной информации чаще уменьшают точность квантования.

Главным критерием допустимых потерь является визуальная оценка изображения и субъективная оценка слышимости аудиоданных.

https://habrahabr.ru/post/251417/

Большинство современных методов удаления визуально избыточной информации используют сведения об особенностях человеческого зрения. Всем известна различная чувствительность человеческого глаза к информации о цветности и яркости изображения. На рисунке показано изображение в цветовом пространстве, закодированное с разной глубиной квантования цветоразностных сигналов:



Глубина квантования цветоразностных сигналов может быть понижена с 256 до 32 уровней с минимальными визуальными изменениями. В то же время потери в цветности весьма существенны и показаны га рисунках.

23.4 ***Программные средства сжатия данных***

Для сжатия данных в файлах существуют специальные программы, называемые **архиваторами**. Эти программы сжимают файлы, значительно уменьшая их размер.

«Классическими» форматами сжатия данных, являются форматы *.ZIP, *.ARJ, *.RAR.

Операционная система	Формат сжатия	Средство архивации	Средство разархивации
MS-DOS	.ZIP	PKZIP.EXE	PKUNZIP.EXE
	.ARJ	ARJ.EXE	
	.RAR	RAR.EXE	UNRAR.EXE
Windows	.RAR	WinRAR	
	.ZIP	WinZip	
	.ARJ	WinArj	
RED OS	.ZIP	7-ZIP	

Современные программные средства для создания и обслуживания архивов отличаются большим объёмом функциональных возможностей, многие из которых выходят за рамки простого сжатия данных и эффективно дополняют стандартные средства операционной системы. В этом смысле современные средства архивации данных называют **диспетчерами архивов.**

Требования к диспетчерам архивов

- Извлечение файлов из архивов;
- Создание новых архивов;
- Добавление файлов в имеющийся архив;
- Создание самораспаковывающихся архивов;
- Создание распределённых архивов на носителях малой ёмкости;
- Тестирование целостности структуры архивов;
- Полное восстановление повреждённых архивов;
- Защита архивов от просмотра и несанкционированной модификации.

В случаях, когда архивация производится для передачи документа потребителю, следует предусмотреть наличие у него программного средства, необходимого для извлечения исходных данных из уплотнённого архива. Если таких средств нет, то создаются *самораспаковывающиеся* архивы.

Дополнительные требования к архиваторам

Дополнительные сервисные услуги обеспечивают:

- Просмотр файлов различных форматов без извлечения их из архивов;
- Поиск файлов и данных внутри архивов;
- Проверку отсутствия компьютерных вирусов в архиве до его распаковки;
- Установку программ из архивов без предварительной распаковки.
- Криптографическую защиту архивной информации;
- Создание самораспаковывающихся многотомных архивов.

Самораспаковывающиеся архивы готовятся на базе обычного архива путём присоединения к нему небольшого программного модуля. Архив получает расширение имени *.EXE. Потребитель запускает его как программу, после чего распаковка произойдёт автоматически.

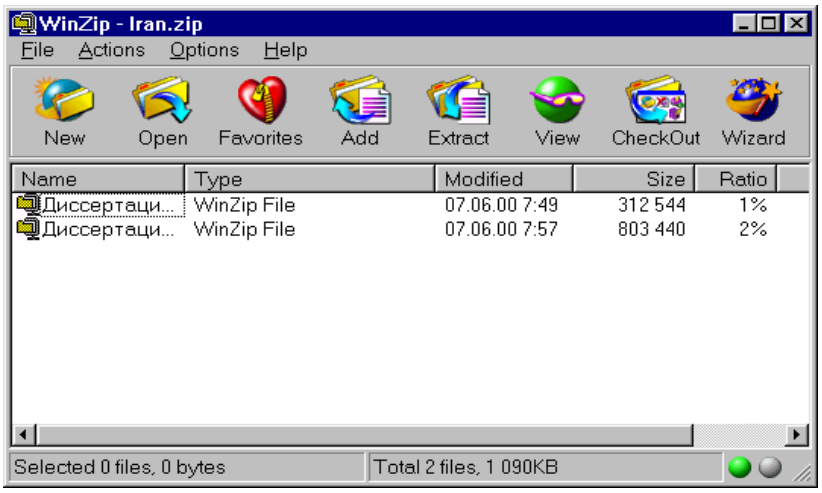
Распределённые архивы организуются в случае передачи большого архива на носителях малой ёмкости (на гибких дисках). Возможно распределение одного архива в виде малых фрагментов на нескольких носителях.

Архиватор WinZip выполняет разбиение архива сразу на гибкие диски.

При этом каждый том несёт файл с одинаковым именем. Например, Инф.Zip.

Нельзя установить № тома по названию файла. Каждый диск следует маркировать пометками на наклейке. Чтобы узнать № тома следует вызвать диалоговое окно *Свойства,* на вкладке *Общие* в поле *Метка* можно узнать № тома.

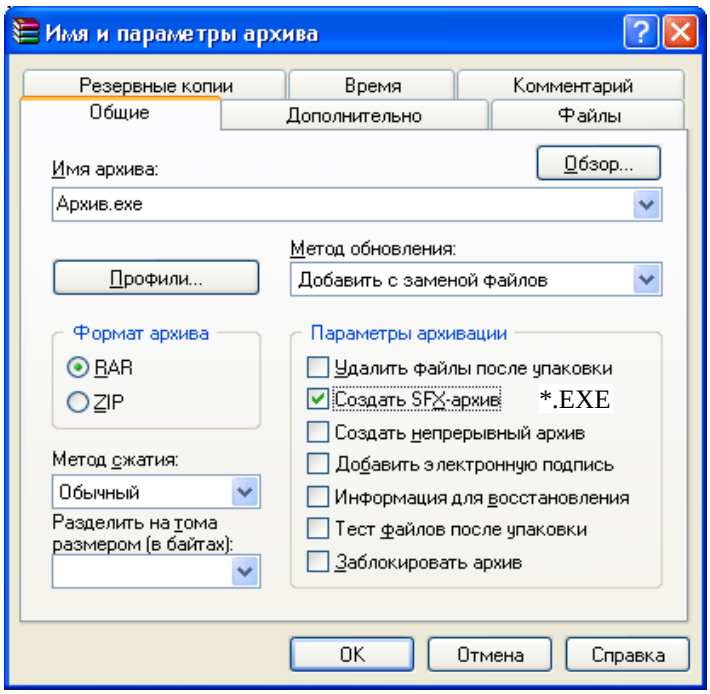
Интерфейс диспетчера архива WinZip



Панель инструментов

Размер исходного файла
Размер сжатого файла
Коэффициент сжатия

Интерфейс диспетчера архива WinRar



Архиваторы WinRar и WinArj позволяют выполнить предварительное разбиение архива на фрагменты заданного размера на жестком диске. Затем их можно перенести на дискету путём копирования. Эти архиваторы маркируют все файлы распределённого архива разными именами.

Например,
Инф01.Rar
Инф02.Rar
Инф03.Rar
и т.п.

WinRar позволяет осуществлять сжатие объектов как в формате .Rar так и формате .Zip, изменять методы сжатия от обычного до быстрого, максимального, скоростного.

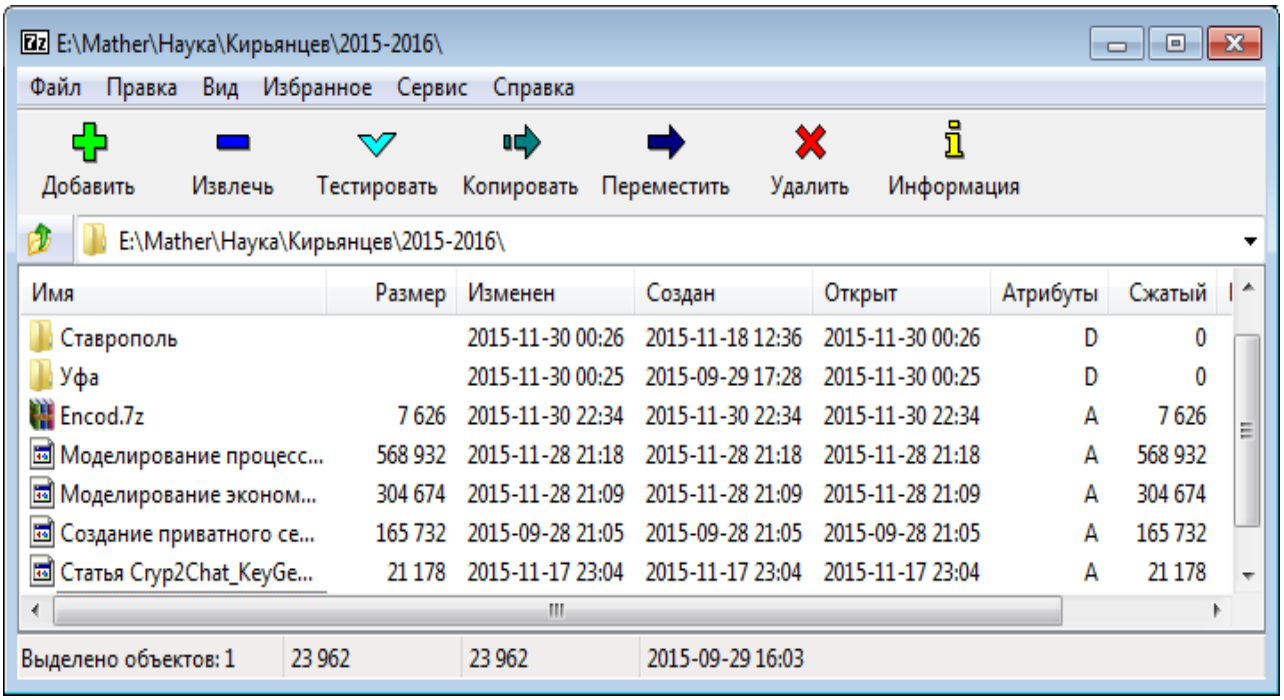
Программа 7-Zip File Manager

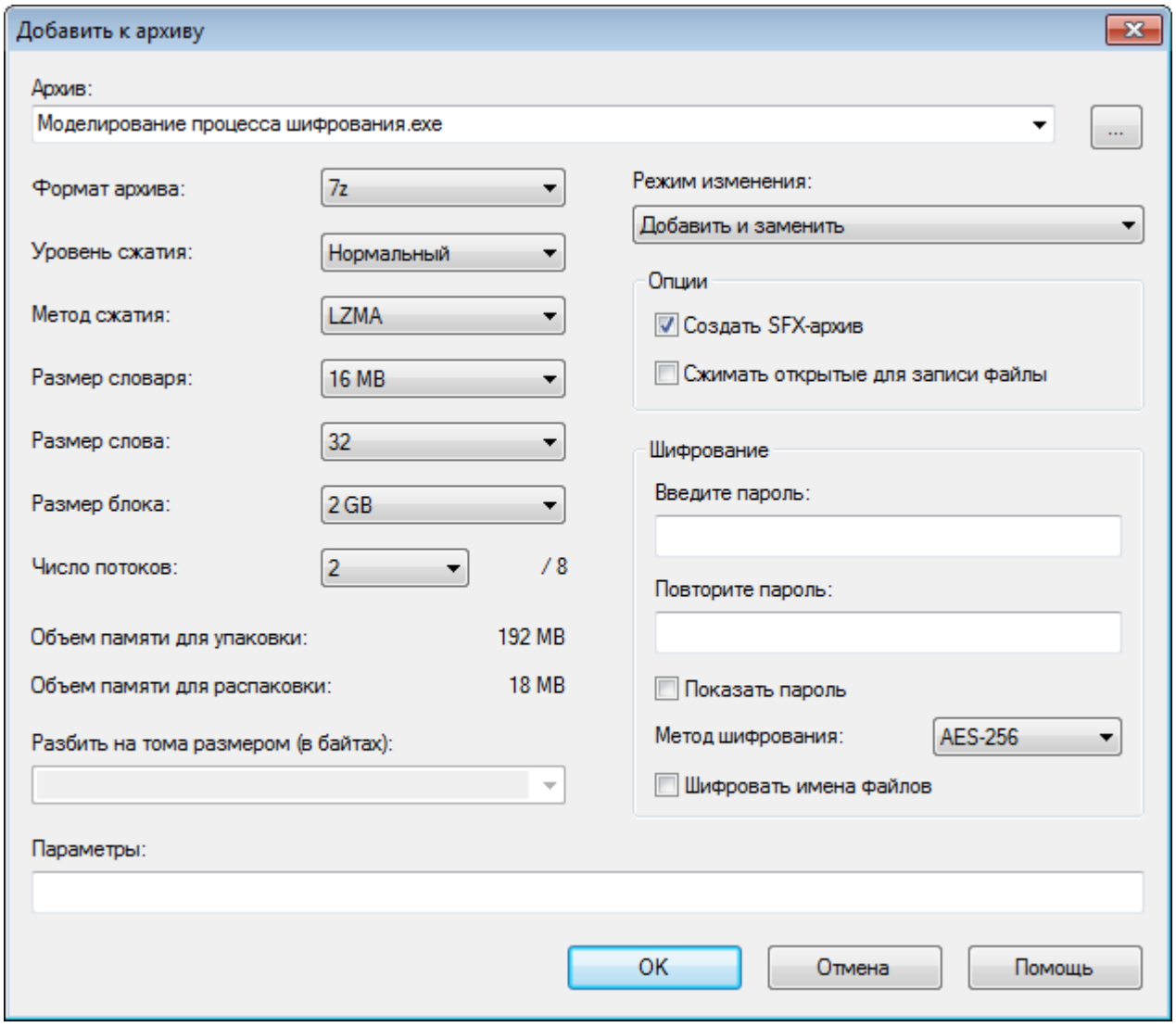
Архивный файл создается в следующей последовательности:

1. Выделяется файл, подлежащего архивации (или несколько пиктограмм).
2. Для выделенного объекта вызвать контекстное меню и выбрать в нём команду *7-Zip / Добавить к архиву...*.
3. В поле *Архив* окна диалога будет прописано имя архивного файла. При необходимости это имя можно заменить другим.
4. В списке *Уровень сжатия* окна установить *Нормальный* (или любой другой).
5. При необходимости можно выбрать методы шифрования и сжатия, а так же размер словаря.

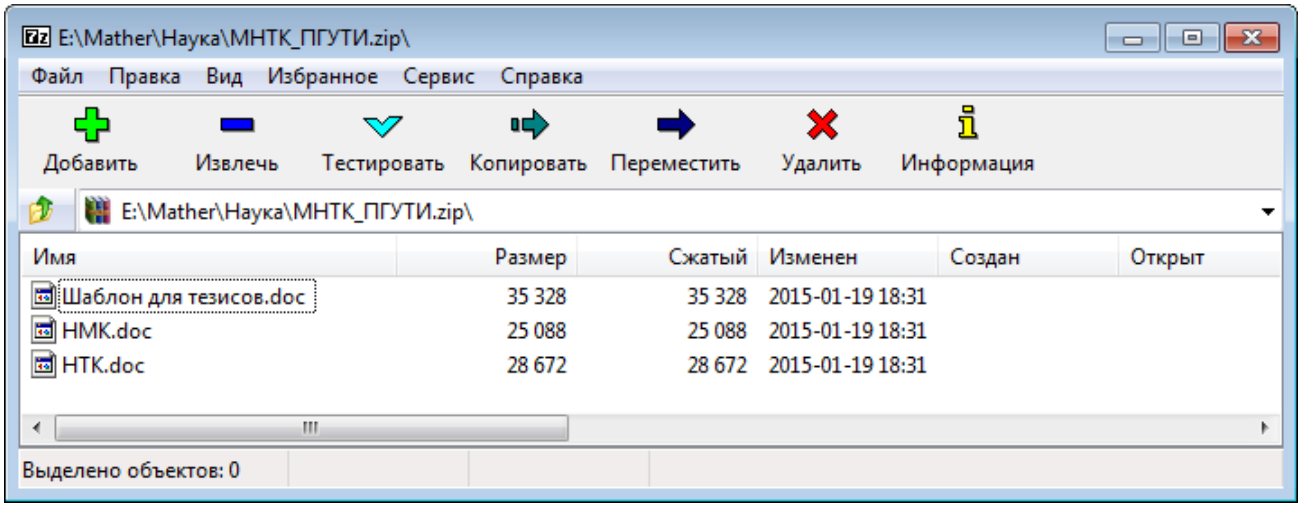
6. Щёлкнуть по кнопке *ОК* окна ***Добавить к архиву***.

Интерфейс Программа 7-Zip File Manager





Извлечь файлы из архива можно по одноименной команде



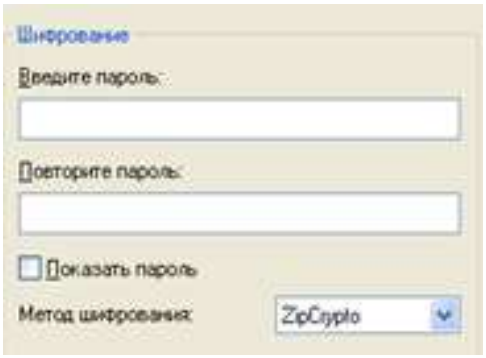
Извлечение файла создается в следующей последовательности:

1. Выделяется пиктограмма файла, подлежащего разархивации.
2. Для выделенного объекта вызвать контекстное меню и выбрать в нём команду *7-Zip / Распаковать.*
3. В открывшемся окне ***Извлечь*** в поле *Распаковать в*: с помощью кнопки *Обзор папок* установить на дереве папок путь к папке, в которую необходимо поместить исходный файл.
4. Щёлкнуть по кнопке *ОК.*

Защита архивов

Защиту архивов выполняют с помощью пароля, который запрашивается при попытке просмотреть, распаковать или изменить архив.

Защита с помощью пароля считается неудовлетворительной и не рекомендуется для особо важной информации.



В название пароля лучше вводить:

- сочетание символов русского и английского языка,
- знаки препинания, цифры,

(чтобы опровергнуть попытки снять пароль путём переборов символов английского алфавита). Чтобы не забыть пароль со временем, вводимые в пароль символы, должны ассоциироваться с какими-то датами, событиями вашей жизни, и т.п.