

# Информация и информационные процессы

## § 3. Сжатие данных

# Что такое сжатие?

Алфавит: **A, B, C, \_**

Сообщение: **ABA CABA**



80 битов в 8-битной кодировке!

**A** → 00      **C** → 10

**B** → 01      **\_** → 11

**ABA CABA** → 00 01 00 11 10 00 01 00 01 00

20 битов



Как раскодировать?

## Словарь:

	00	01	10	11
	00000100 <sub>2</sub>	01000001 <sub>2</sub>	01000010 <sub>2</sub>	01000011 <sub>2</sub>
	4 символа	<b>A</b> (код 65)	<b>B</b> (код 66)	<b>C</b> (код 67)
				<b>пробел</b> (код 32)

## Коэффициент сжатия

---

Сообщение: **10240 символов**

Алфавит: **A, B, C, \_**

Словарь: **5 байтов**

Длина кода:

$$10240 \times 2 = 20480 \text{ битов} = \mathbf{2560 \text{ байтов}}$$

Длина сжатого сообщения:

$$5 + 2560 = \mathbf{2565 \text{ байтов}}$$

**Коэффициент сжатия** – это отношение размеров исходного и сжатого файлов.

$$k = \frac{10240}{2565} \approx 4$$

# Сжатие без потерь

**Сжатие без потерь** – это такое уменьшение объема закодированных данных, при котором можно восстановить их исходный вид из кода без искажений.



За счёт чего сжимается сообщение?



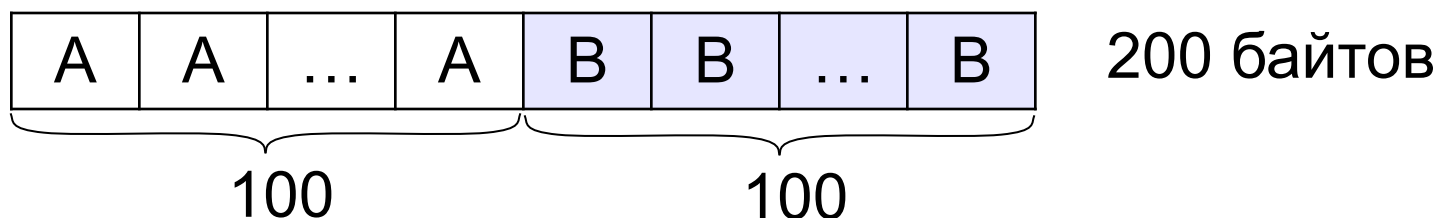
В данных должна быть избыточность!

используются только  
4 символа из 256

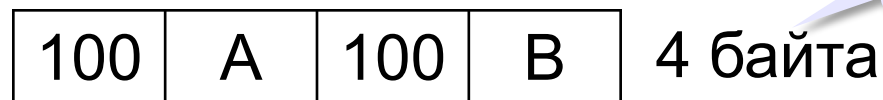
# Алгоритм RLE

**RLE** (англ. *Run Length Encoding*, кодирование цепочек одинаковых символов)

Файл qq.txt



Файл qq.rle (сжатый)



сжатие в 50 раз!



В чем состоит избыточность?

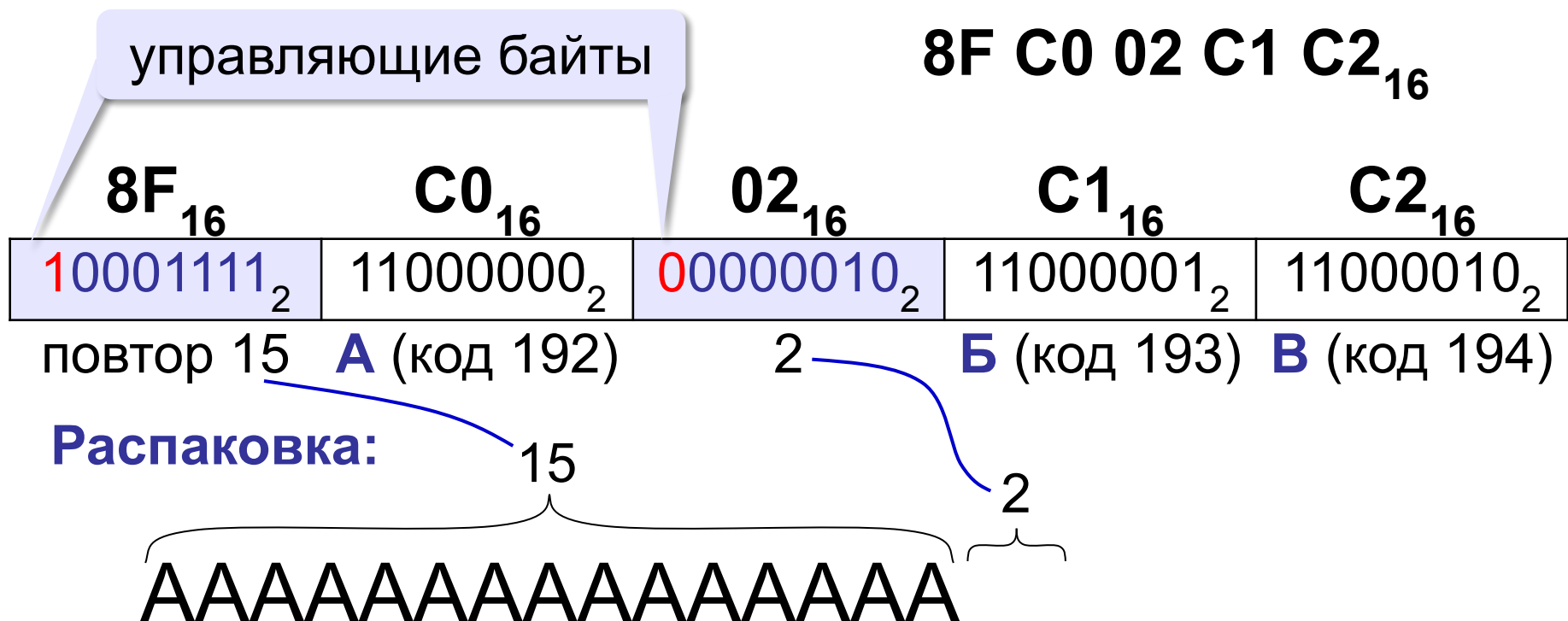


Сжатие с потерями или без?



Что в худшем случае?

# Алгоритм RLE



## Применение:

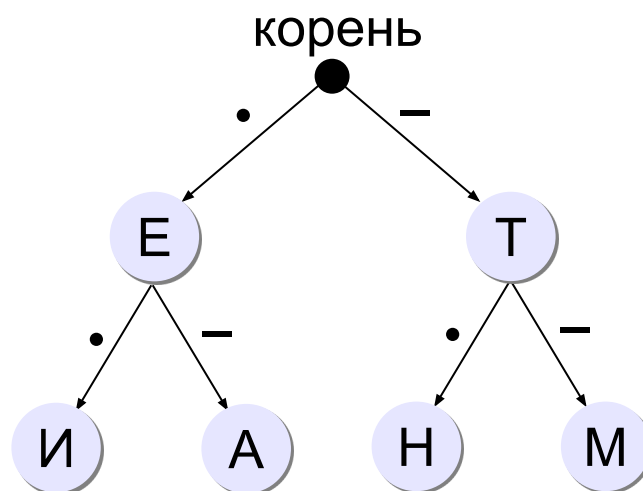
- сжатие рисунков \* .bmp (с палитрой)
- один из этапов сжатия рисунков \* .jpg

# Неравномерные коды

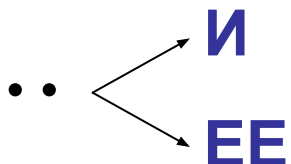
**Идея:** кодировать часто встречающиеся символы более короткими кодовыми словами.

**Азбука Морзе:**

<b>Е</b>	•	<b>И</b>	••
<b>Т</b>	—	<b>А</b>	•—
		<b>Н</b>	—•
		<b>М</b>	— —



Проблема: разделить последовательность на кодовые слова!

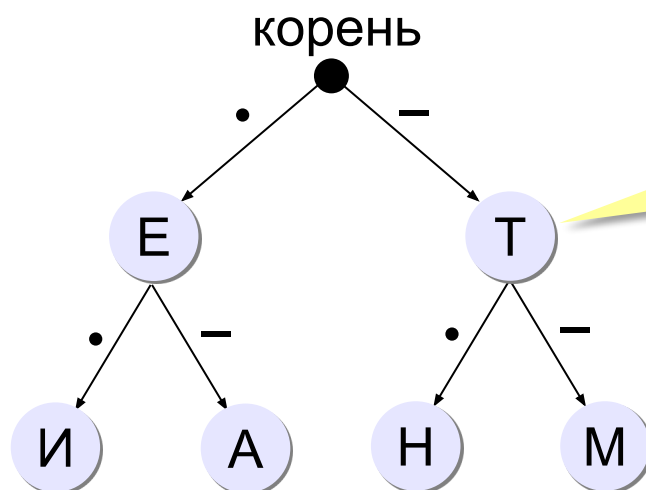


Можно ли обойтись без разделителя?

# Префиксные коды

**Префиксный код** – это код, в котором ни одно кодовое слово не является началом другого кодового слова (условие Фано).

Е	•	И	••
Т	–	А	•–
		Н	–•
		М	––



НЕ ВСЕ СИМВОЛЫ  
В ЛИСТЯХ!



Это не префиксный код!



Проблема: как построить префиксный код?



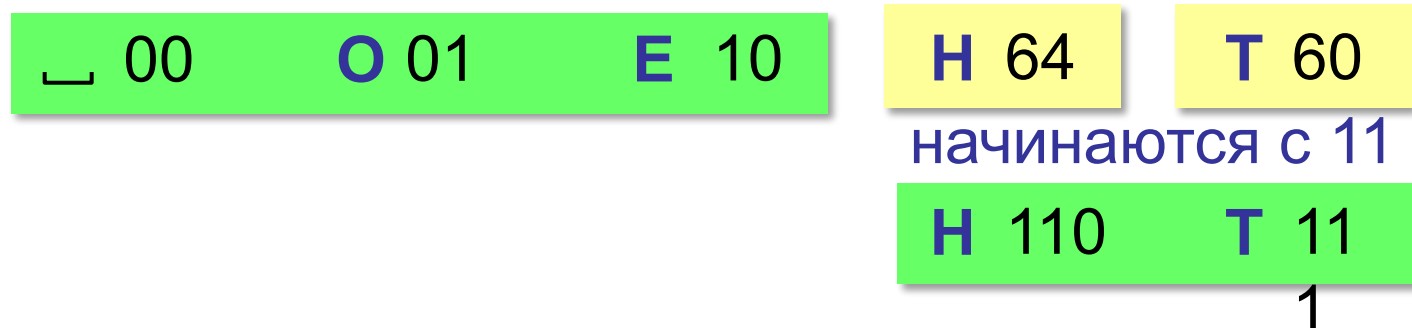
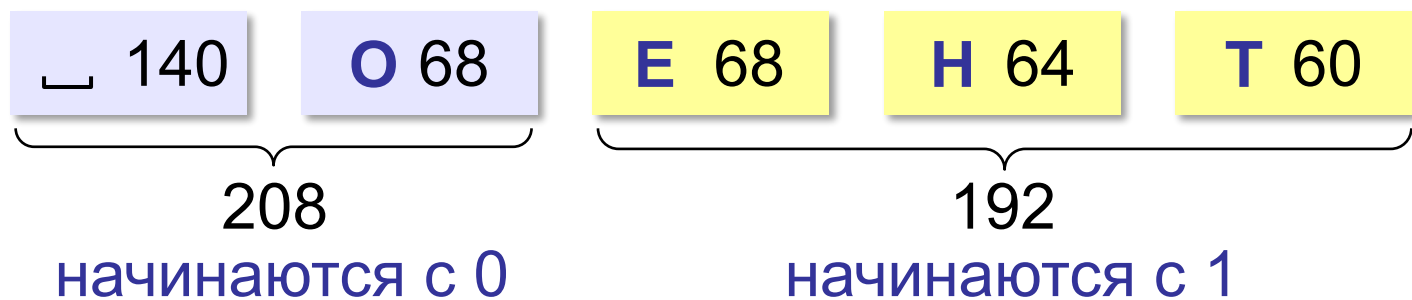
# Код Шеннона-Фано

Алфавит: **О**, **Е**, **Н**, **Т**, **␣**

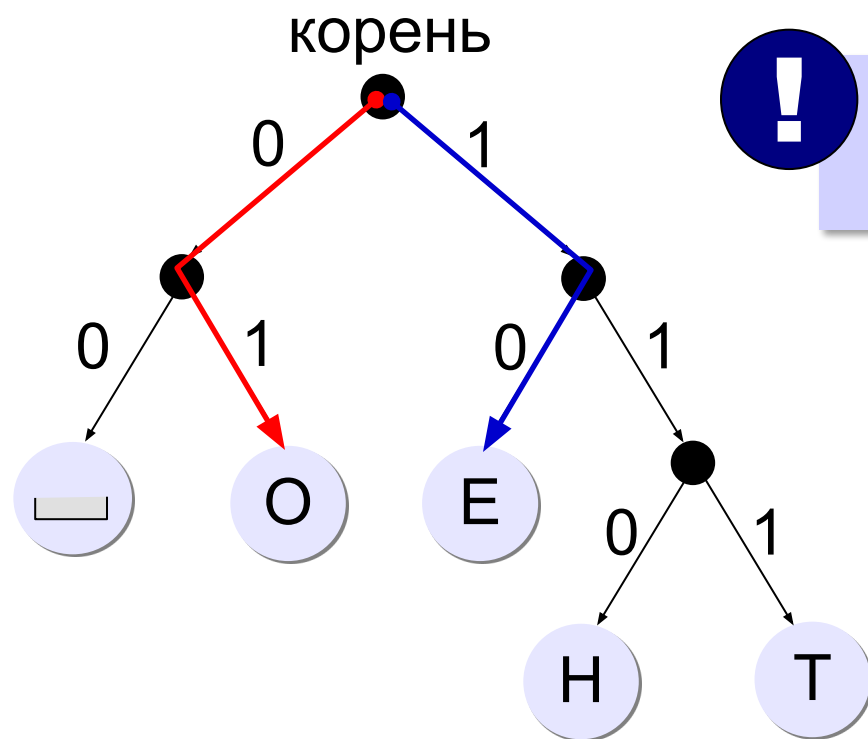
Количество символов в сообщении:



На 2 группы с примерно равным числом символов:



# Код Шеннона-Фано



Это префиксный код (все символы в листьях дерева)!

Декодирование:

1110111101001011001111  
 1T O 1T O \_ E H O 1T

# Код Шеннона-Фано

---



- учитывается частота символов
- не нужен символ-разделитель
- код префиксный – можно декодировать по мере поступления данных



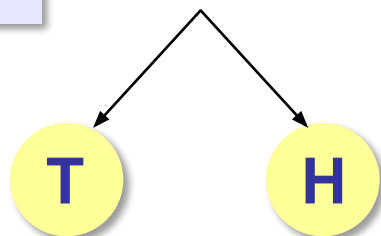
- нужно заранее знать частоты символов
- код неоптимален
- при ошибке в передаче сложно восстановить «ХВОСТ»
- не учитывает повторяющиеся последовательности СИМВОЛОВ

# Алгоритм Хаффмана

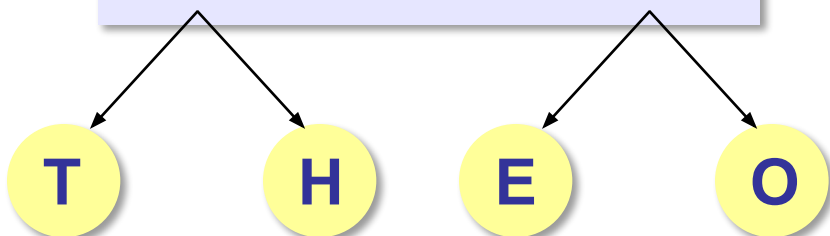
По увеличению частоты:

**Т** 60    **Н** 64    **Е** 68    **О** 68     $\lfloor$  140

**Е** 68    **О** 68    124     $\lfloor$  140

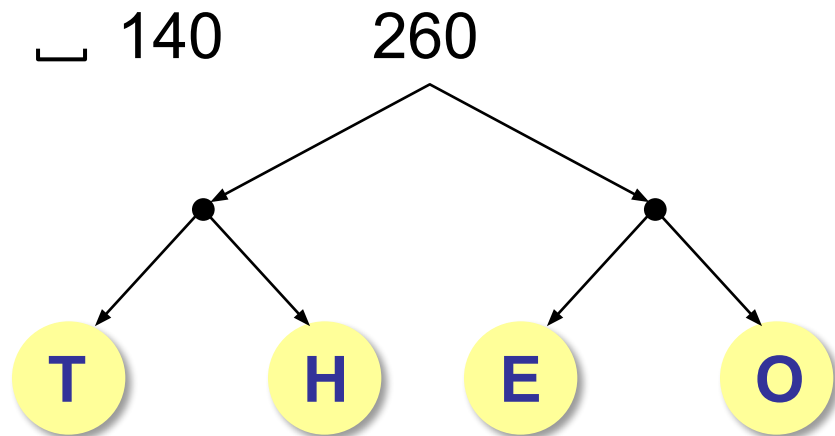


124    136     $\lfloor$  140



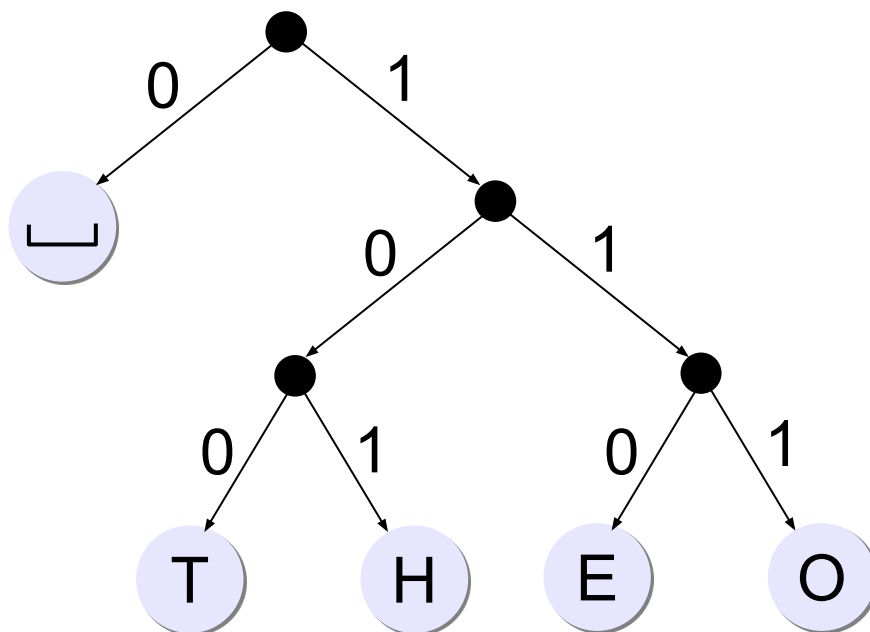
Дэвид Хаффман

# Алгоритм Хаффмана



**Код Хаффмана:**

└ 0	<b>Т</b> 100	<b>Н</b> 101
	<b>Е</b> 110	<b>О</b> 11
		1



# Сравнение алгоритмов

---

Количество символов в сообщении:

Л 140    О 68    Е 68    Н 64    Т 60

**Равномерное кодирование (8-битный код):**

$$(140 + 68 + 68 + 64 + 60) \cdot 8 = 3200$$

БИТОВ

**Равномерное кодирование (3-битный код):**

$$(140 + 68 + 68 + 64 + 60) \cdot 3 = 1200$$

БИТОВ

+ словарь!



В чём избыточность?

# Сравнение алгоритмов

Количество символов в сообщении:

Л 140    О 68    Е 68    Н 64    Т 60

**Код Шеннона-Фано:**

Л 00    О 01    Е 10    Н 110    Т 11

$$(140 + 68 + 68) \cdot 2 + (64 + 60) \cdot 3 = 924 \quad 1$$

бита

$$k = \frac{1200}{924} \approx 1,299$$

**Код Хаффмана:**

Л 0    О 11    Е 110    Н 101    Т 100

$$140 + (68 + 68 + 64 + 60) \cdot 3 = 920$$

бит

$$k = \frac{1200}{920} \approx 1,304$$



Оптimalен!

# Алгоритм Хаффмана

---



- код оптимальный среди алфавитных кодов



- нужно заранее знать частоты символов

- при ошибке в передаче сложно восстановить «ХВОСТ»

- не учитывает повторяющиеся последовательности СИМВОЛОВ




# Алгоритм LZW

---

1977: А. Лемпел и Я. Зив, 1984: Т. Велч

## Идеи:

- кодировать не отдельные символы, а блоки
  - последовательностям символов присваиваются числовые коды
  - новая цепочка  $\Rightarrow$  занесение в словарь с новым кодом
- 
  - словарь строится по мере получения данных
  - не нужны частоты символов  $\Rightarrow$  за один проход!

## Применение:

- сжатие рисунков `*.gif`, `*.tif`
- сжатие документов `*.pdf`

# Сжатие с потерями

**Сжатие с потерями** – это такое уменьшение объема закодированных данных, при которых распакованный файл может отличаться от оригинала.

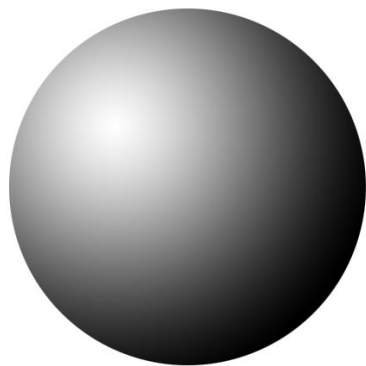
**Идея:** «отбросить» часть данных, которые не влияют на восприятие информации человеком (доп. размытие фотографий, частоты выше 20 кГц, ...)

## Применение:

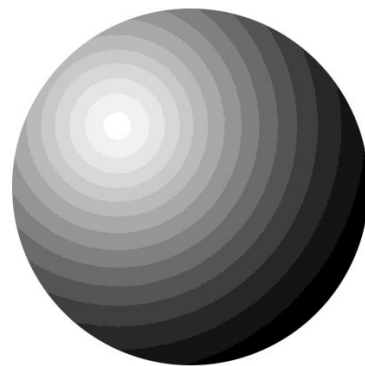
- сжатие рисунков \* .jpg, \* .jpeg
- сжатие звука \* .mp3, \* .aac, \* .ogg, ...
- сжатие видео \* .mpg, \* .wmv, \* .mov, ...

# Снижение глубины цвета

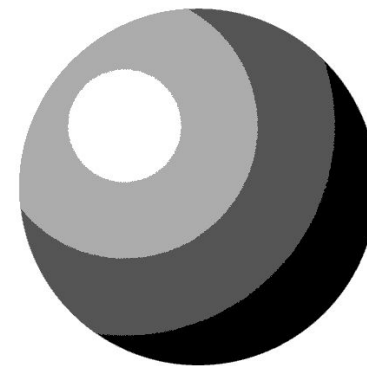
---



8 битов на пиксель  
(256 цветов)



4 бита на пиксель  
(16 цветов)



2 бита на пиксель  
(4 цвета)



**размер** ↓

**качество** ↓

# Сжатие JPEG

яркость

«синева»

**RGB** → **Y Cb Cr**

«краснота»

$Y = 0,299 \cdot R +$  глаз чувствительнее к зелёному!

$Cb = 128 - 0,1687 \cdot R - 0,3313 \cdot G + 0,5$

$\cdot B$

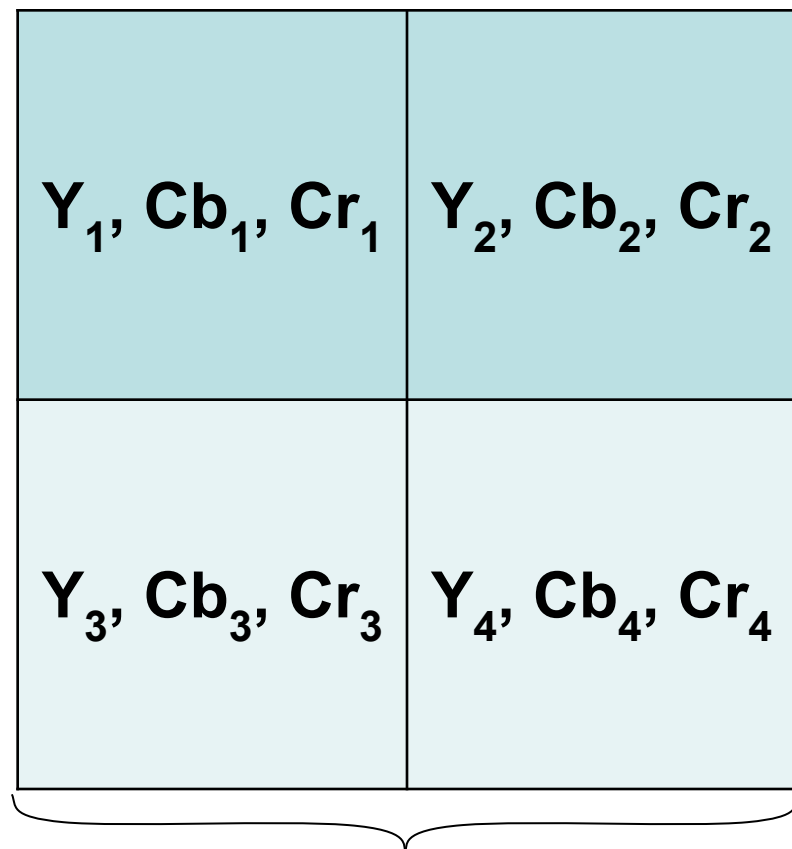
$Cr = 128 + 0,5 \cdot R - 0,4187 \cdot G - 0,0813$

**?** Что для чёрно-белого (серого)?

$Cb = Cr = 128$

# Сжатие JPEG

**Идея:** глаз наиболее чувствителен к яркости



**12 чисел**

**6 чисел**

$\Rightarrow Y_1, Y_2, Y_3, Y_4, Cb, Cr$   
например:

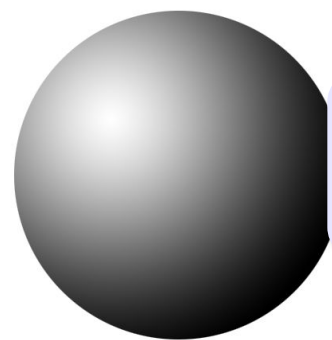
$$Cb = \frac{Cb_1 + Cb_2 + Cb_3 + Cb_4}{4}$$

$$Cr = \frac{Cr_1 + Cr_2 + Cr_3 + Cr_4}{4}$$

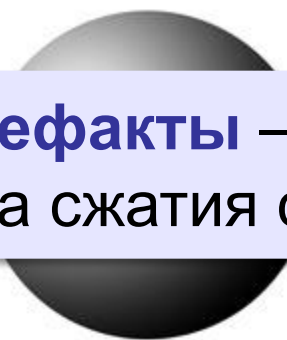
потери!

+ *дискретное косинусное преобразование*, алгоритмы RLE и Хаффмана

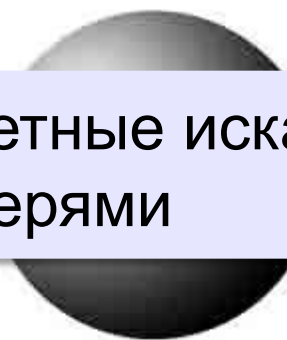
# Сжатие JPEG



качество 100  
(8400 байтов)



качество 50  
(3165 байтов)

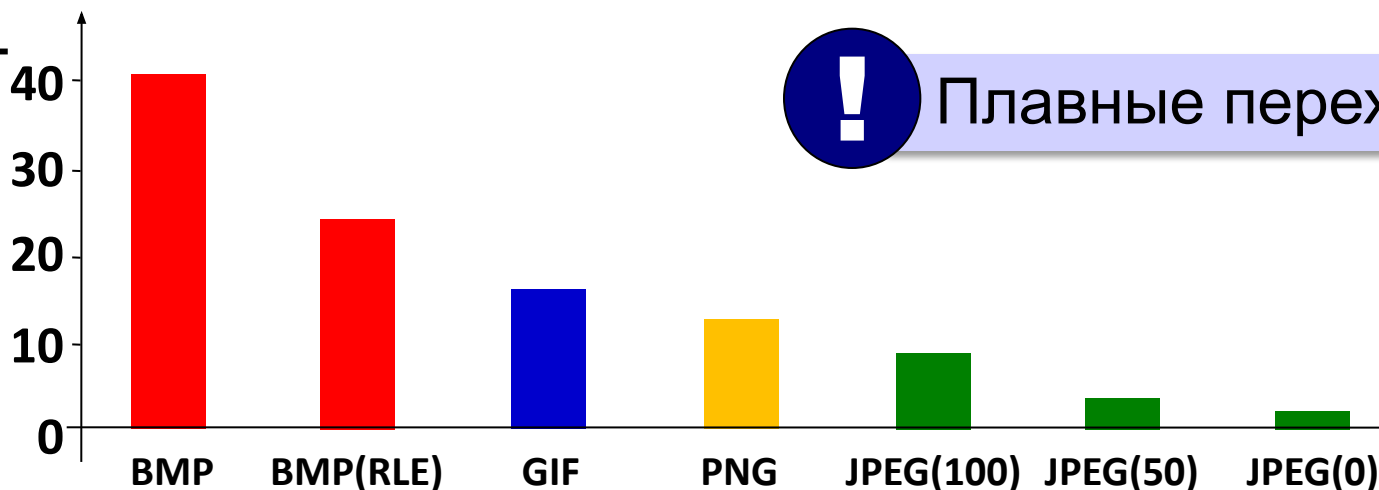


качество 0  
(1757 байтов)



**Артефакты** – заметные искажения из-за сжатия с потерями

V,  
Кбайт



Плавные переходы!

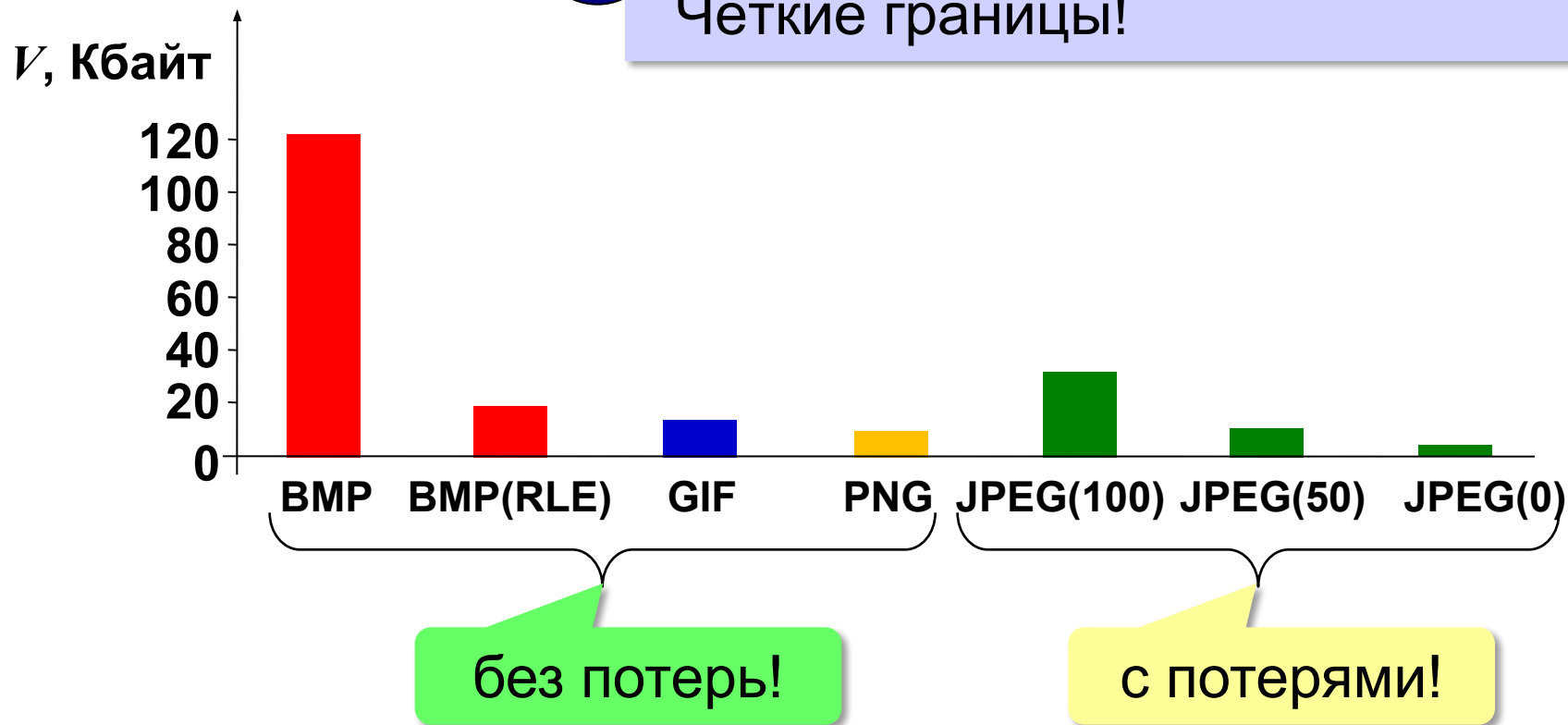
# Сжатие рисунков с потерями и без



Что особенного?



Большие области одного цвета!  
Чёткие границы!



без потерь!

с потерями!

## Сжатие звука (MP3)

**MP3** = MPEG-1 Layer 3, кодирование восприятия

**Битрейт** – это число бит, используемых для кодирования 1 секунды звука.

MP3: от 8 до 320 кбит/с

Без сжатия на CD (1 сек, 44 кГц, 16 бит, стерео):

$2 \times 88000 = 176\,000$  байт =  $1\,408\,000$  бит = **1408 кбит**

Сжатие MP3 (**256 кбит/с**):

$$k = \frac{1408}{256} \approx 5,5$$



# Сжатие видео

---

видео = изображения + звук

**Кодек** (кодировщик/декодировщик) – это программа для сжатия данных и восстановления сжатых данных.

MJPEG, MPEG-4, DivX, Xvid, H.264, ...



**Артефакты** – заметные искажения из-за сжатия с потерями

# Сжатие: итоги

---



Сжатие уменьшает избыточность данных!

## Хорошо сжимаются:

- тексты (\*.txt)
- документы (\*.doc)
- несжатые рисунки (\*.bmp)
- несжатый звук (\*.wav)
- несжатое видео (\*.avi)



Нужно ли стремиться к полному удалению избыточности?

## Плохо сжимаются:

- случайные данные
- сжатые данные в архивах (\*.zip, \*.rar, \*.7z)
- сжатые рисунки (\*.jpg, \*.gif, \*.png)
- сжатый звук (\*.mp3, \*.aac)
- сжатое видео (\*.mpg, \*.mp4, \*.mov)