

# Информатика. Введение.

Лекция 2

Доцент каф. АСУ, к.т.н.:  
Суханов А.Я.

## Понятие информации

Информация – одно из самых фундаментальных понятий в современной науке, наряду с веществом, энергией, пространством и временем.

Фундаментальное, то есть первичное понятие невозможно строго определить через вторичные или производные понятия – это основа всего научного познания.



## **Информация в быту**

Под **информацией в быту** понимают любые сведения об окружающем мире и протекающих в нем процессах, воспринимаемые человеком (с помощью органов слуха, зрения, осязания, обоняния, вкуса) или специальными устройствами.

# Информация в технике

Под **информацией в технике** понимают любые сообщения, которые зафиксированы в виде знаков и могут передаваться в виде сигналов.

# Информация в теории управления

**Под информацией в теории управления (менеджменте)** понимают сообщения, уменьшающие существующую до этого неопределенность в той предметной области, к которой они относятся, и использующиеся для совершения активного действия, например, управленческого решения.

# ключевые атрибуты информации.

1. **Достоверность.** информация свободна от ошибок, чьей-либо пристрастности и отражает истинное положение дел. Часто организации применяют независимые источники информации, чтобы анализируя их, уменьшать фактор пристрастности в принимаемом решении или в распространяемой производной информации.
2. **Оперативность.** Доставка информации получателям в рамках необходимых временных границ. Например, вчерашняя газета сегодня, запоздавшая котировка акций. Своевременность просто означает, что адресат должен получить информацию, когда ему нужно.
3. **Актуальность**, т.е. важность, существенность для настоящего времени. Точная и своевременная информация может в то же время быть неактуальной, более того информация, актуальная для одного получателя, не обязательно актуальна для другого.
4. **Полнота.** Информация должна содержать все важные данные, которые ожидают от нее пользователи, и ее должно быть достаточно для понимания и принятия решения.
5. **Полезность.** Полезность (ценность) информации определяется по тем задачам, которые можно решить с ее помощью.
6. **Понятность** означает, что информация может быть представлена в ясном и понятном для потребителя формате. Потребитель информации – лицо, принимающее решение, должен как можно меньше времени тратить на дополнительные уточнения поступившей информации.

## Информация в узком и широком смыслах

В узком смысле информацией можно назвать сведения о предметах, фактах, понятиях некоторой предметной области.

С середины XX века **информация** рассматривается в **широком смысле** как общенавучное понятие, включающее в себя как совокупность сведений об объектах и явлениях окружающей среды, их параметрах, свойствах и состоянии, так и обмен сведениями между людьми, человеком и автоматом, автоматом и автоматом, обмен сигналами между живой и неживой природой, в животном и растительном мире, а также генетическую информацию.

## Виды информации

информацию можно подразделить на:

- 1) **структурную** (или связанную) присущую объектам неживой и живой природы естественного или искусственного происхождения. Эти объекты (орудия труда, предметы быта, произведения искусства, научные теории и т.п.) возникают путем опредмечивания циркулирующей информации, то есть благодаря и в результате целенаправленных управленческих процессов;
- 2) **оперативную** (или рабочую), циркулирующую между объектами материального мира и используемую в процессах управления в живой природе, в человеческом обществе

## Данные, знания

*Сведения, полученные путем измерения, наблюдения, логических или арифметических операций, и представленные в форме, пригодной для постоянного хранения, передачи и обработки получили название **данные**.*

*Совокупность полезной информации, правил и процедур ее обработки, необходимая для получения новой информации о какой-либо предметной области называют **знанием**.*

# Свойства знаний

1. **Внутренняя интерпретируемость** знаний (понятность знания его носителю).
2. **Структурированность знаний.** Информационные единицы должны обладать гибкой структурой. Принцип «матрешки» – рекурсивная вложимость знаний. Возможность произвольного установления и перенастройки отношений (включения) между информационными единицами.
3. **Связность.** Отношения между элементами: структурные, функциональные, каузальные и семантические. Структурные задают иерархию, функциональные задают процедурную информацию, позволяющие находить одни элементы через другие, каузальные задают причинно-следственные связи, семантические охватывают все остальные виды отношений.
4. **Ассоциативность знаний** – наличие семантической метрики в сфере знаний. Отношение релевантности на множестве информационных единиц характеризует ситуационную близость элементов (силу ассоциативной связи). Позволяет находить знания, близкие к уже найденным.
5. **Активность знаний** – наличие у знаний побуждающей и направляющей функции, что фактически превращает знания в квазипотребности. Актуализации тех или иных действий способствуют имеющиеся в системе знания.

# Необходимость количественной оценки

Введение информации в научно-технический и хозяйственный оборот привело к потребности в её **количественной оценке**.

Требовалось создать меру сравнения для практического использования информации.

В простейшей комбинаторной форме эта мера была предложена **Р. Хартли в 1928 году** – первый шаг к созданию теории информации.

**Пример 1.** Как определить, какая из двух монет фальшивая, если на вид они одинаковы, но известно, что фальшивая легче. Проводим одно взвешивание на чашечных весах, и все становится ясно. Таким образом, до взвешивания у вас была *неопределенность* по поводу того, какая из монет фальшивая, а после взвешивания вы сняли эту *неопределенность*, получив *информацию*. Иными словами, вы получили сообщение в элементарном альтернативном выборе между двумя событиями («фальшивая – не фальшивая», «да - нет», «истина – ложь», «0–1»).



# Понятие бита

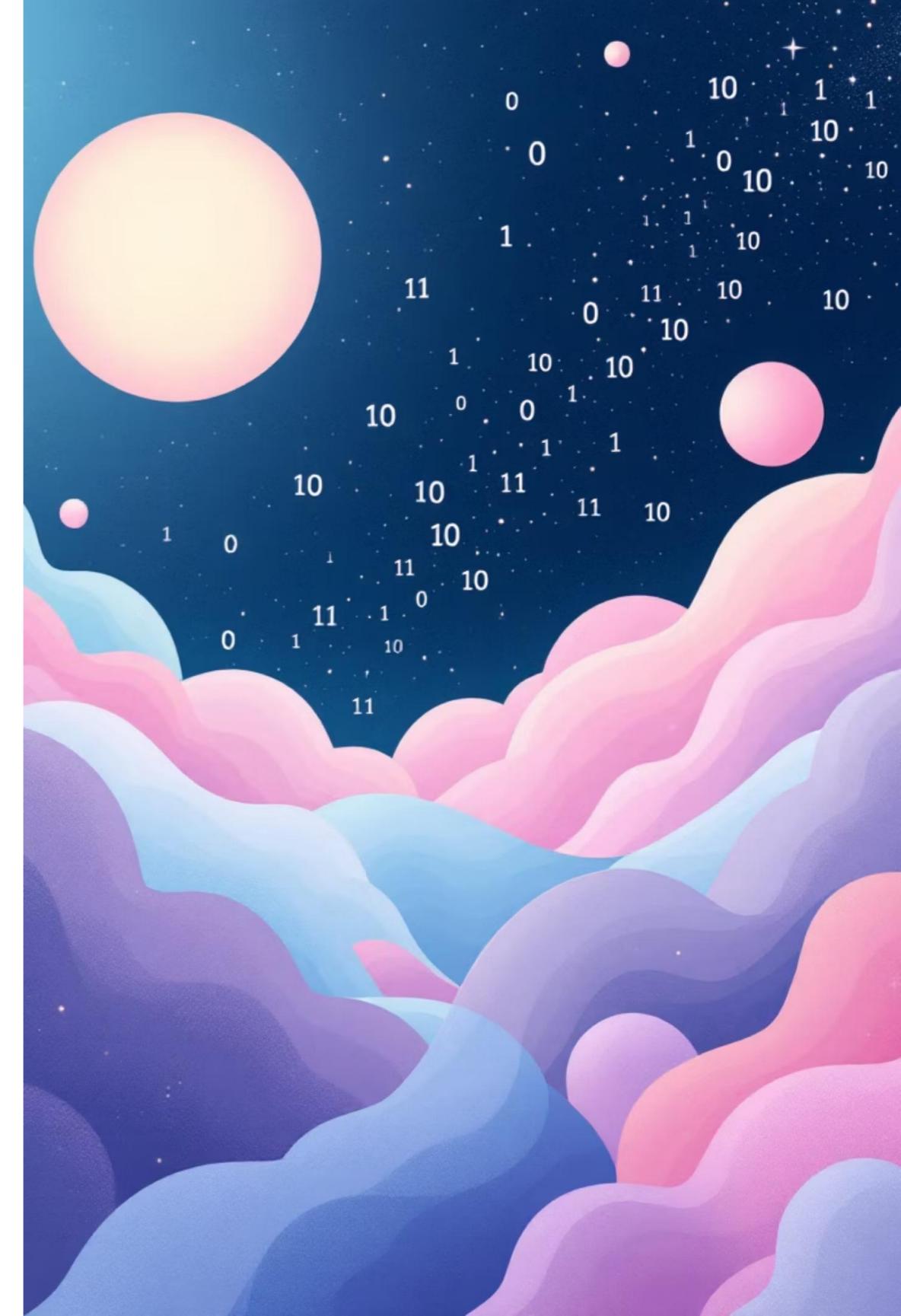
## Двойственная природа

Бит – это одновременно **двоичный знак** и **единица измерения** количества информации

## Определение

Количество информации в выборе с двумя взаимоисключающими равновероятными исходами

Это фундаментальная единица, которая стала основой всей цифровой революции.



**Пример 2.** А если монет 8? Тогда делим их на две равные части и взвешиваем их. Ту часть, которая легче, снова делим на две части и снова взвешиваем и т.д. За три взвешивания мы определим фальшивую монету.

За три выбора мы уменьшили существующую неопределенность в 2, 4, 8 раз, получив таким образом 3 бита информации.

**Пример 3.** Перейдем от монет к картам. Пусть в колоде из 32 карт необходимо угадать определенную карту, например, туза пик. Для этого необходимо и достаточно получить ответы «да» и «нет» на **пять** вопросов. Вопросы, ответы на которые позволяют выбрать одну из альтернатив, называют двоичными, или бинарными. Ответами на эти вопросы мы уменьшаем неопределенность в 2, 4, 8, 16, 32 раз. В конце неопределенности не остается. Количество полученной информации равно 5 бит

Вопрос	Ответ	Бинарный ответ
1.Карта красной масти?	Нет	0
2.Трефы?	Нет	0
3.Одна из четырех старших?	Да	1
4.Одна из двух старших?	Да	1
5.Король?	Нет	0
Значит, задуманная карта была туз пик.		

## Формула Хартли для равновероятных событий

В процессе получения информации мы выбираем одно сообщение из конечного множества  $n$  равновероятных сообщений.

Количество информации  $I$ , содержащееся в выбранном сообщении, определяется как двоичный логарифм  $n$

$$I = \log_2 n$$

Эта закономерность стала основой для развития более сложных теорий измерения информации.



# Вероятности отдельных букв в русском языке (с учетом пробела)

Буква	—	о	е, ё	а	и	т	н	с
$P_i$	0,175	0,090	0,072	0,062	0,062	0,053	0,053	0,045
Буква	р	в	л	к	м	д	п	у
$P_i$	0,040	0,038	0,035	0,028	0,026	0,025	0,023	0,021
Буква	я	ы	з	ь, Ъ	б	г	ч	й
$P_i$	0,018	0,016	0,016	0,014	0,014	0,013	0,012	0,010
Буква	х	ж	ю	ш	ц	щ	э	ф
$P_i$	0,009	0,007	0,006	0,006	0,004	0,003	0,003	0,002

# Частоты букв (в процентах) ряда европейских языков

Буква алфавита	Французский язык	Немецкий язык	Английский язык	Итальянский язык
A	7,68	5,52	7,96	11,12
B	0,80	1,56	1,60	1,07
C	3,32	2,94	2,84	4,11
D	3,60	4,91	4,01	3,54
E	17,76	19,18	12,86	11,63
F	1,06	1,96	2,62	1,15
G	1,10	3,60	1,99	1,73
H	0,64	5,02	5,39	0,83
I	7,23	8,21	7,77	12,04
J	0,19	0,16	0,16	-
K	-	1,33	0,41	-

# Формула Шеннона и понятие вероятности

01

Проблема неравновероятности Определение вероятности

Для неравновероятных процессов американский учёный **Клод Шеннон** предложил новую формулу (1948 г.)

02

Численная мера достоверности события при большом числе испытаний:  
 $P = m/n$

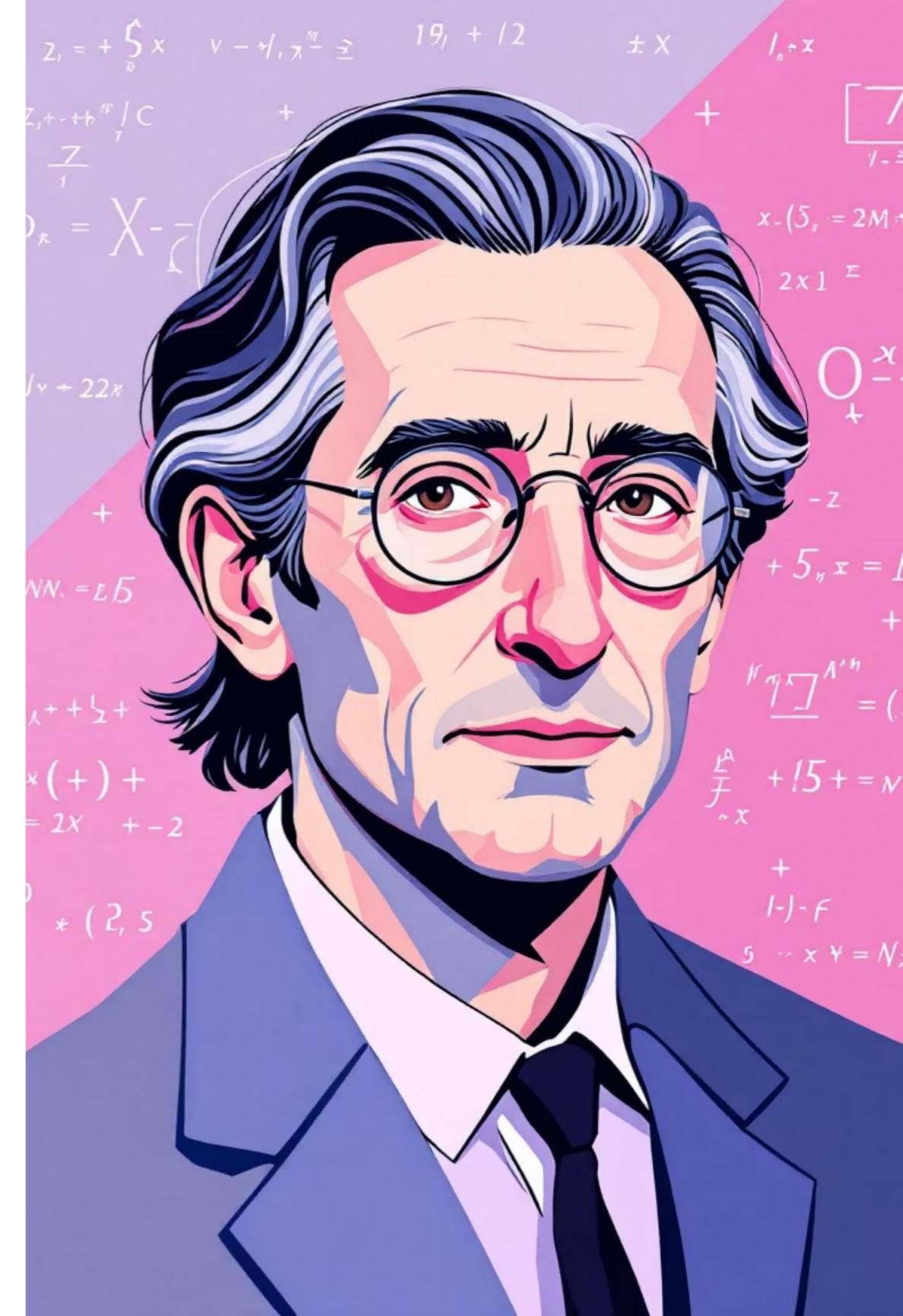
$$P_i = \lim_{n \rightarrow \infty} (m / n)$$

03

Свойства вероятности

Любая вероятность принадлежит отрезку  $[0;1]$ . Событие с  $P=0$  невозможно, с  $P=1$  – достоверно

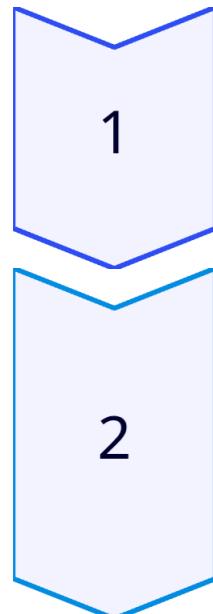
- **Пример:** При многократном подбрасывании монетки орёл выпадает примерно в половине случаев, поэтому  $P(\text{орёл}) = 0,5$  или 50%



# Мера Шеннона и информационная энтропия

Чтобы получить определённый символ от источника сообщений, нужно перебрать  $n = 1/p$  символов, где  $p$  – вероятность появления символа.

Количество двоичных переборов:  
 $\log_2(n) = \log_2(1/p) = -\log_2(p)$



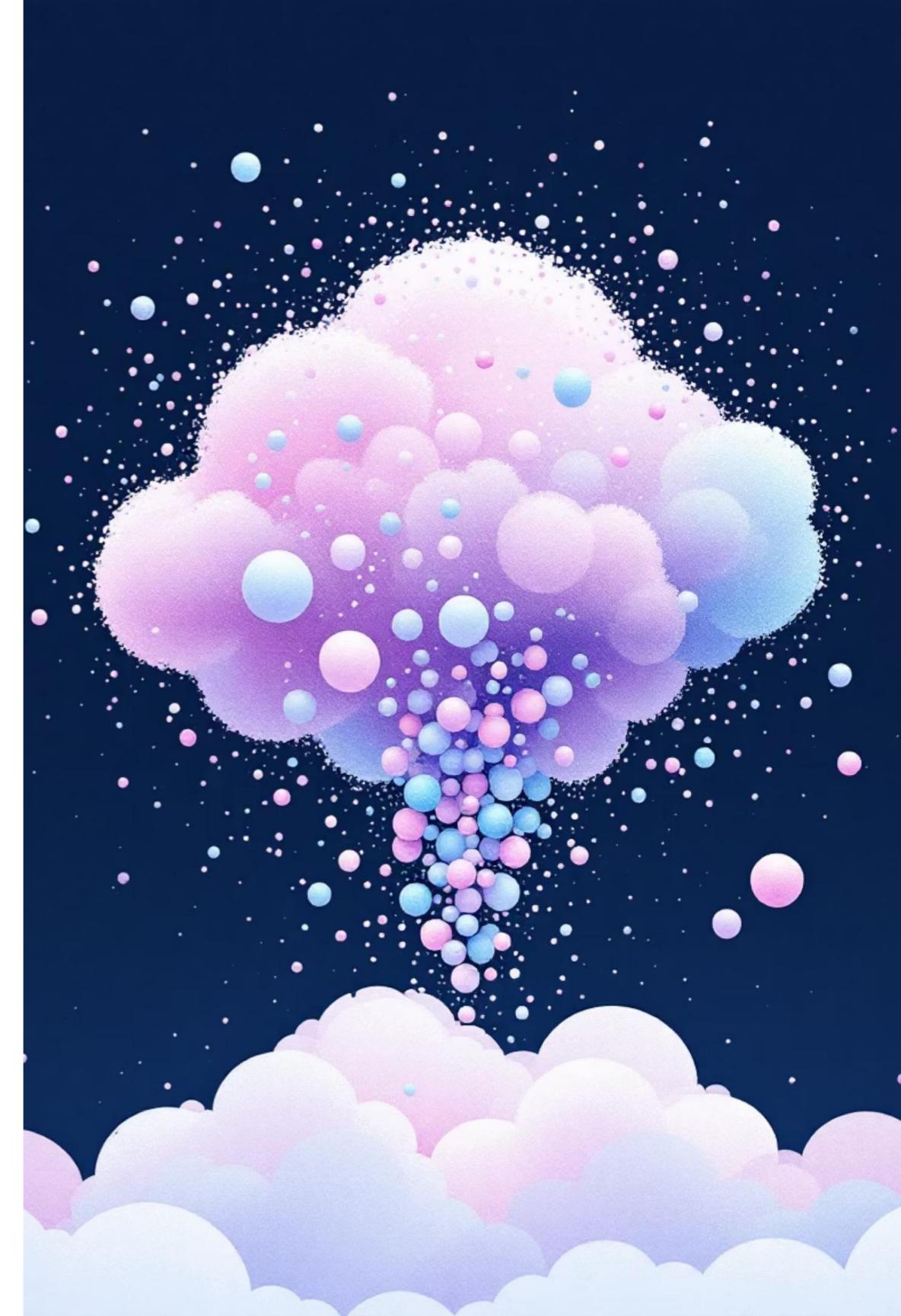
Для одного символа

$$I_i = -\log_2 p_i$$

Формула Шеннона

$$H = - \sum_{i=1}^n p_i \log_2 p_i$$

Эта величина называется **информационной энтропией** и характеризует неопределённость множества сообщений.



## Свойства энтропии и связь с информацией



### Известное сообщение

Энтропия заранее известного сообщения равна 0



### Неизвестные сообщения

Во всех других случаях  $H > 0$

Чем больше энтропия системы, тем больше степень её неопределённости

$$I = H_{\text{до}} - H_{\text{после}}$$

Количество информации измеряется тем, насколько понизилась энтропия системы после поступления сообщения.

- ❑ **Философский смысл:** Уменьшая энтропию, мы получаем информацию – в этом заключается суть научного познания!

# Совместная и условная энтропия

## Совместная энтропия

$$H(X, Y) = -\sum \sum p(x, y) \log_2 p(x, y)$$

Измеряет неопределённость пары случайных величин

- Независимые величины  
Если  $X$  и  $Y$  **независимы**, то  $H(X|Y) = H(X)$  — знание  $Y$  ничего не даёт

- Полная зависимость  
Если  $X$  **полностью определяется**  $Y$  (например,  $X = Y$ ), то  $H(X|Y) = 0$  — неопределённости нет

## Условная энтропия

$$H(X | Y) = -\sum \sum p(x, y) \log_2 p(x | y)$$

Неопределённость  $X$  при известном  $Y$

- Эти концепции лежат в основе современных методов сжатия данных, машинного обучения и криптографии

## Пример с монетой и кубиком

Рассмотрим эксперимент: бросаем честную монету  $X$  и одновременно кидаем игральный кубик  $Y$ . Если события независимы, то знание результата монеты не влияет на результат кубика.

При  $X = \text{орёл}$  условная энтропия кубика составляет:

$$H(Y|X=\text{орёл}) = \log_2 6 \approx 2.585 \text{ бит}$$

Неопределённость остаётся полной, поскольку события независимы:

$$H(Y|X) = H(Y)$$

Пусть:

$Y$  : погода — «дождь» или «солнечно» (с вероятностями 0.5 каждое),

$X$  : взять ли зонт — зависит от погоды:

Если дождь  $\rightarrow$  берём зонт с вероятностью 1,

Если солнечно  $\rightarrow$  берём зонт с вероятностью 0.

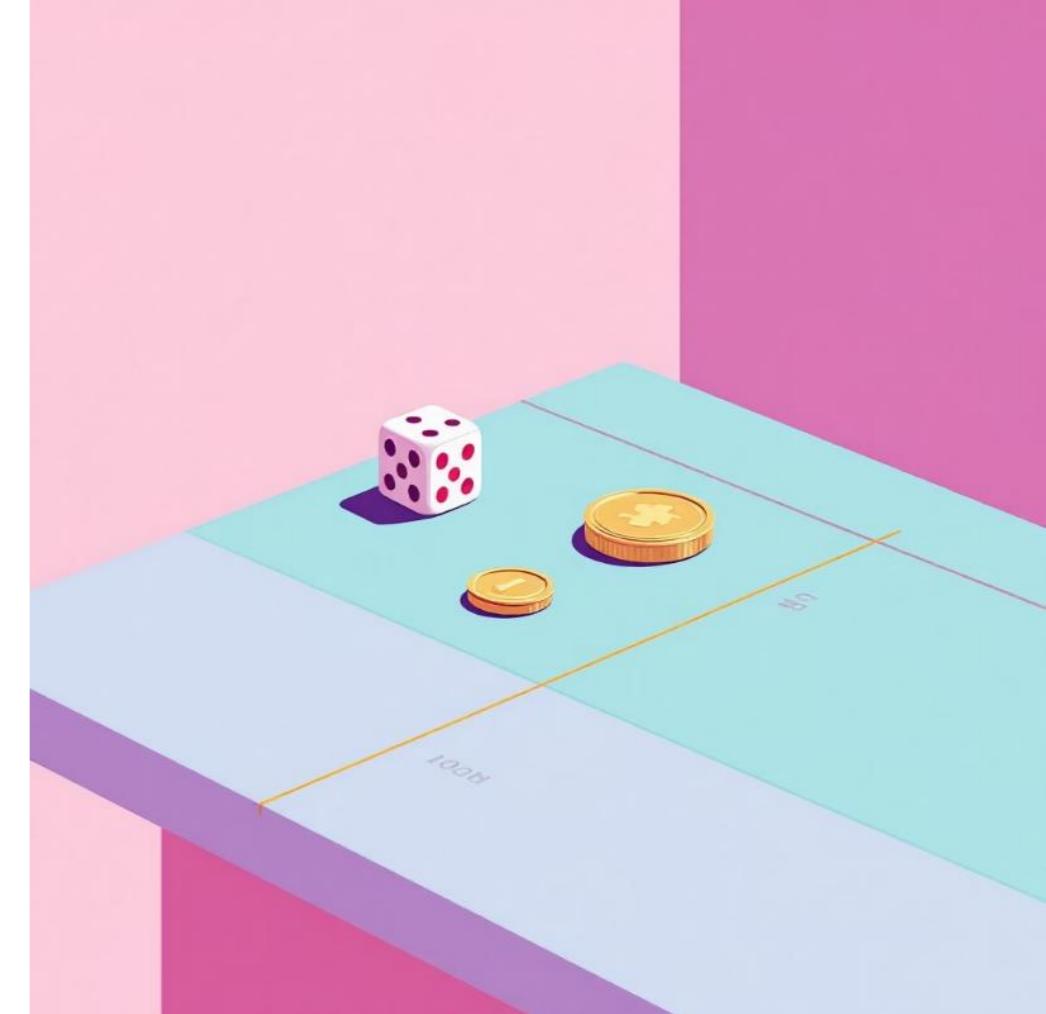
Тогда:

$H(X|Y=\text{дождь})=0$  (всегда берём зонт),

$H(X|Y=\text{солнечно})=0$  (никогда не берём),

$$\Rightarrow H(X|Y)=0.5 \cdot 0 + 0.5 \cdot 0 = 0 .$$

Знание погоды **полностью устраниет неопределённость** в решении о зонте.



# Взаимная информация: сколько информации о Y даёт знание X

1

## Основное определение

Взаимная информация  $I(X; Y)$  количественно измеряет, сколько информации знание случайной величины  $X$  предоставляет о случайной величине  $Y$ , и наоборот. Это фундаментальная мера зависимости между переменными.

2

## Основная формула

$$I(X; Y) = H(Y) - H(Y | X) = H(X) - H(X | Y)$$

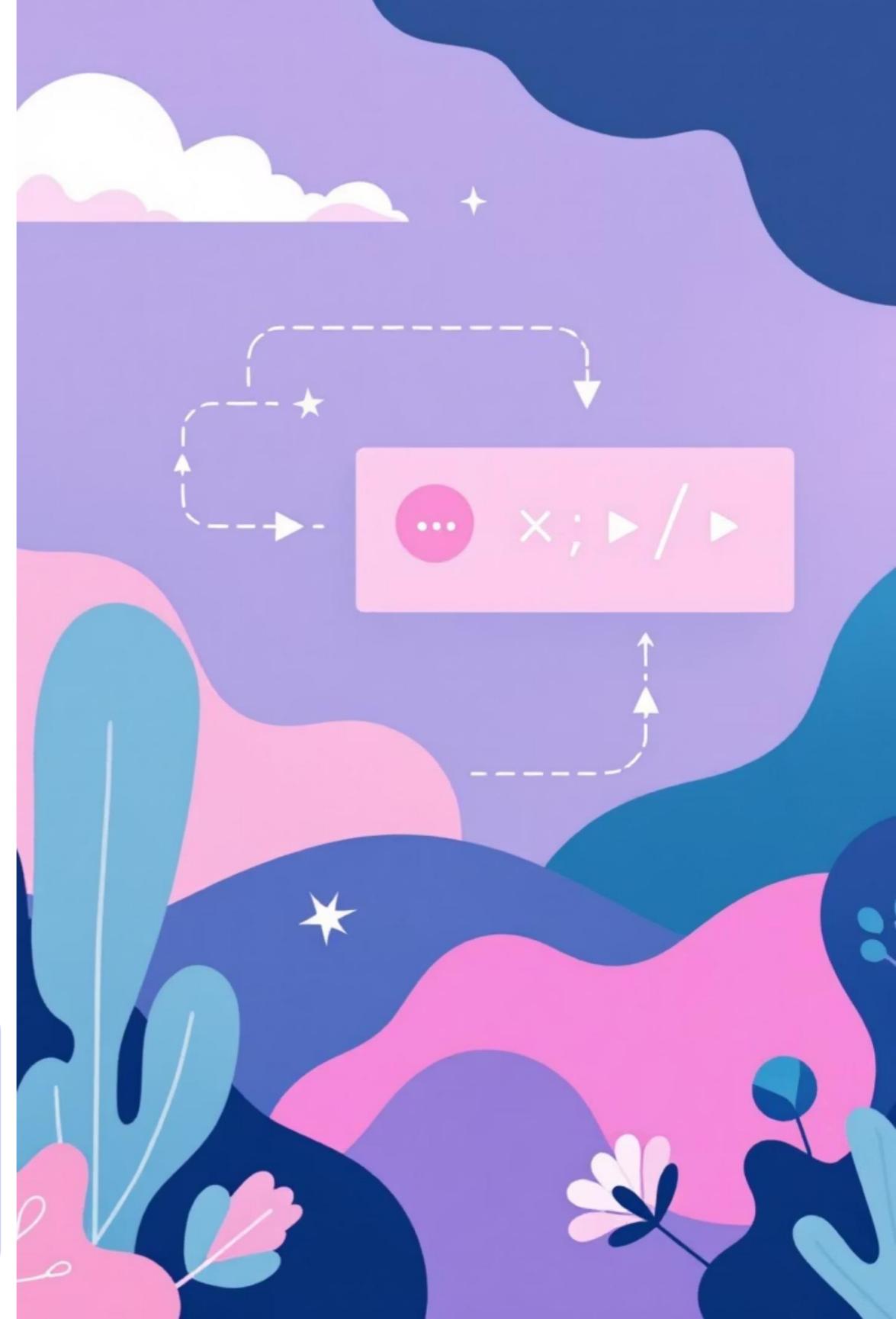
Альтернативная запись через совместную энтропию:

$$I(X; Y) = H(X) + H(Y) - H(X, Y)$$

## Практический пример с зависимыми событиями

- Пусть  $X$  — результат броска честной монеты, а  $Y$  — результат броска кубика с особым правилом: если монета показывает орла, кубик может показать только чётные числа (2, 4, 6).

В этом случае  $I(X; Y) > 0$ , поскольку знание результата монеты существенно уменьшает неопределённость о результате кубика — с 6 возможных исходов до 3.



Мера **количества информации**, которую одна случайная величина содержит о другой случайной величине. Показывает, **на сколько уменьшается неопределенность** в  $X$ , когда известно  $Y$

**Простая аналогия:** Вы пытаетесь угадать, какая сегодня погода, зная только температуру воздуха. Взаимоинформация показывает, насколько знание температуры помогает угадать погоду.

$$I(X; Y) = \sum \sum p(x,y) * \log( p(x,y) / (p(x)*p(y)) )$$

Пусть:

$Y$  : погода — 0 (солнечно) или 1 (дождь),  $P(Y=0)=P(Y=1)=0.5$  ; солнечная или дождливая погода в вероятностью 0.5

$X$  : решение — 0 (не брать зонт) или 1 (взять зонт).

Если солнечно → не берём зонт:  $P(X=0|Y=0)=1$  ;

Если дождь → берём зонт:  $P(X=1|Y=1)=1$  .

Тогда совместное распределение:

$P(0,0)=0.5$ ,  $P(1,1)=0.5$  , остальные вероятности — 0.

Энтропии:

$H(X)=H(Y)=1$  бит,

$H(X|Y)=0$  (погода полностью определяет решение),

Взаимная информация:  $I(X;Y)=H(X)-H(X|Y)=1$  бит.

Знание погоды полностью устраняет неопределенность в решении о зонте.



### Практические применения

- Оценка зависимости признаков в машинном обучении
- Оптимизация алгоритмов сжатия данных
- Анализ биологических и коммуникационных сигналов
- Построение решающих деревьев

# Кодирование источника сообщений

ТУСУР  
TUSUR UNIVERSITY

Как уже отмечалось, результат одного отдельного альтернативного выбора может быть представлен как 0 или 1. Тогда выбору всякого сообщения (события, символа т.п.) в массиве сообщений соответствует некоторая последовательность двоичных знаков 0 или 1, то есть двоичное слово. Это двоичное слово называют кодировкой, а множество кодировок источника сообщений – кодом источника сообщений.

# Кодирование – замена информационного слова на кодовое

*Пример.*

Информационное слово	Кодовое слово
000	0000
001	0011
010	0101
011	0110
100	1001
101	1010
110	1100
111	1111

Если количество символов представляет собой степень двойки ( $n = 2^N$ ) и все знаки равновероятны  $P_i = (1/2)^N$ , то все двоичные слова имеют длину  $L=N=Id(n)$ . Такие коды называют **равномерными кодами**.

Более оптимальным с точки зрения объема передаваемой информации является **неравномерное кодирование**, когда разным сообщениям в массиве сообщений назначают кодировку разной длины. Причем, часто происходящим событиям желательно назначать кодировку меньшей длины и наоборот, т.е. учитывать их вероятность.

## Кодирование словами постоянной длины

Буква	$a$	$b$	$c$	$d$	$e$	$f$	$g$
Кодирование	000	001	010	011	100	101	110

$$Id(7) \approx 2,807 \text{ и } L=3.$$

Проведем кодирование, разбивая исходное множество знаков на равновероятные подмножества, то есть так, чтобы при каждом разбиении суммы вероятностей для знаков одного подмножества и для другого подмножества были одинаковы. Для этого сначала расположим знаки в порядке уменьшения их вероятностей

Символ	Вероятность, $P_i$	Кодировка	Длина, $L_i$	Вероятность×Длина, $P_i \times L_i$
$a$	0,25	00	2	0,5
$e$	0,25	01	2	0,5
$f$	0,125	100	3	0,375
$c$	0,125	101	3	0,375
$b$	0,125	110	3	0,375
$d$	0,0625	1110	4	0,25
$g$	0,0625	1111	4	0,25

В общем случае **алгоритм построения оптимального кода**

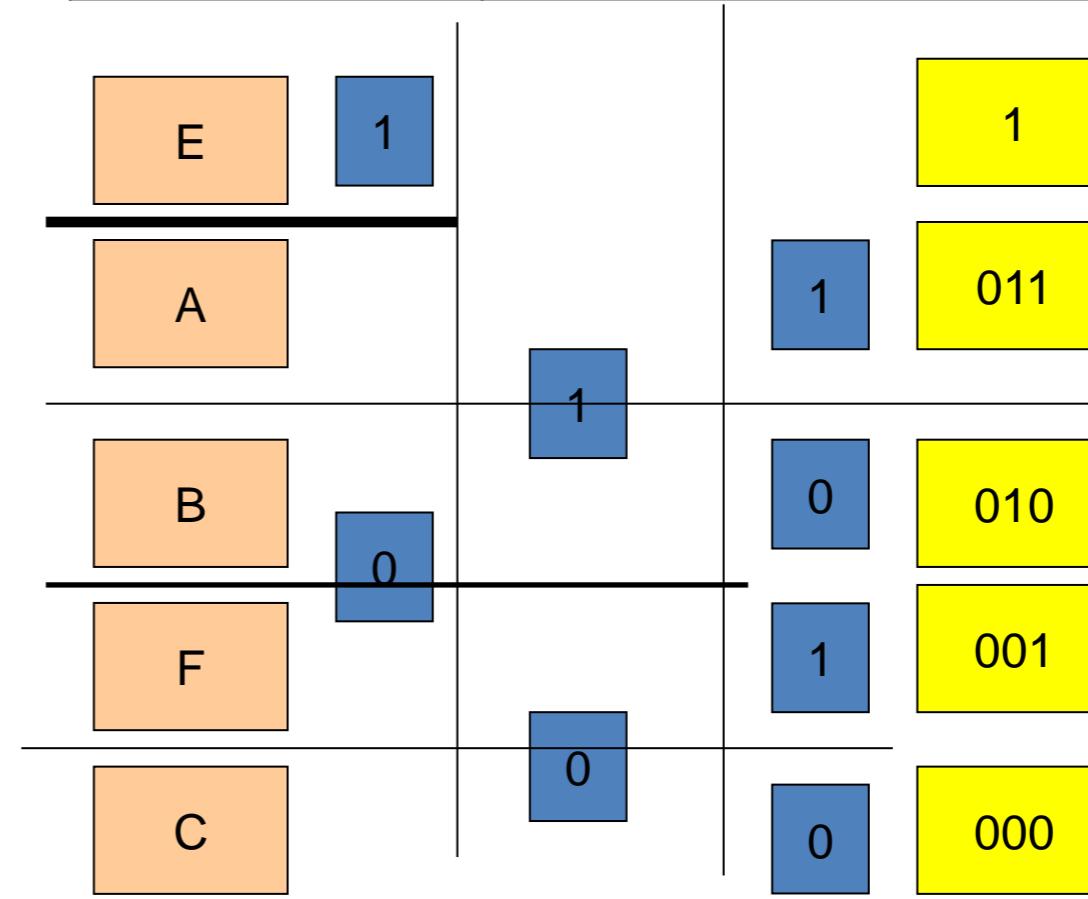
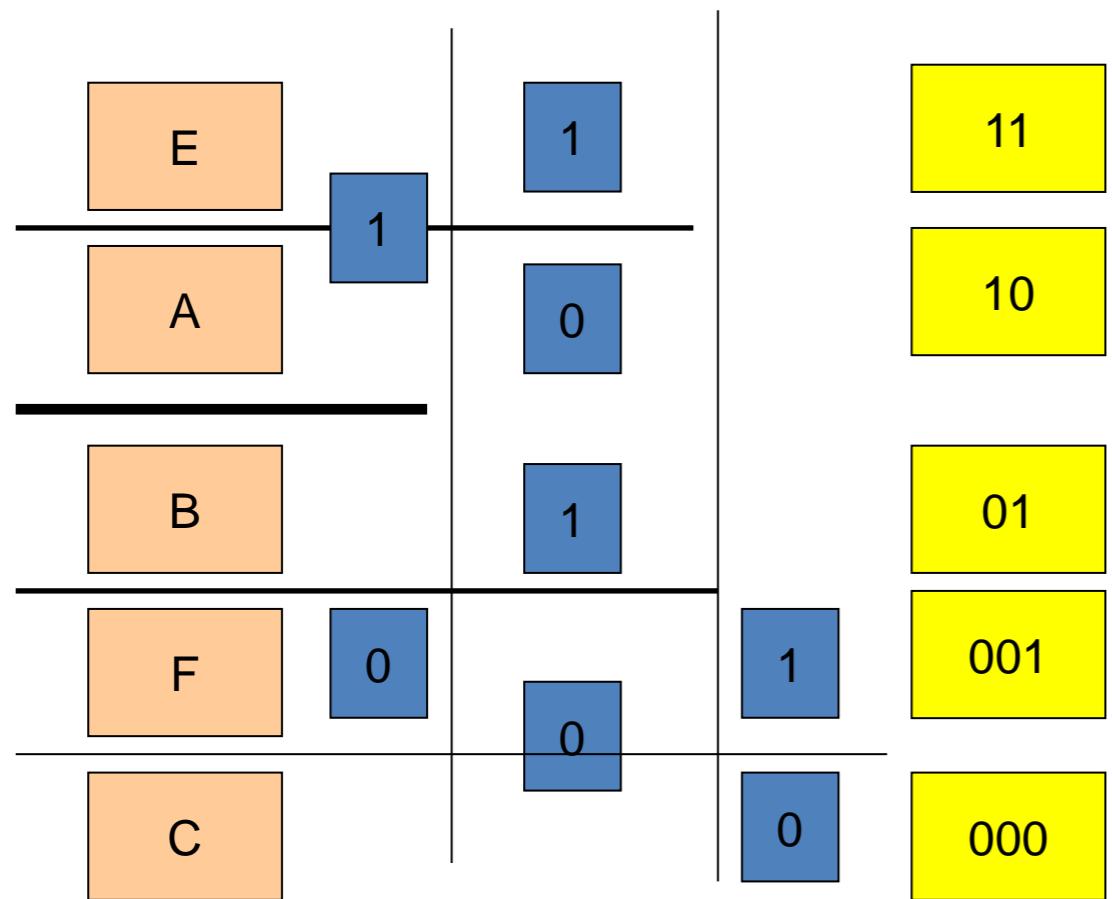
**Шеннона-Фано** выглядит следующим образом:

1. сообщения, входящие в ансамбль, располагаются в столбец по мере убывания вероятностей;
2. выбирается основание кода  $K$  (в нашем случае  $K=2$ );
3. все сообщения ансамбля разбиваются на  $K$  групп с суммарными вероятностями внутри каждой группы как можно близкими друг к другу.
4. всем сообщениям первой группы в качестве первого символа присваивается 0, сообщениям второй группы – символ 1, а сообщениям  $K$ -й группы – символ  $(K-1)$ ; тем самым обеспечивается равная вероятность появления всех символов 0, 1, ...,  $K$  на первой позиции в кодовых словах;
5. каждая из групп делится на  $K$  подгрупп с примерно равной суммарной вероятностью в каждой подгруппе. Всем сообщениям первых подгрупп в качестве второго символа присваивается 0, всем сообщениям вторых подгрупп – 1, а сообщениям  $K$ -ых подгрупп – символ  $(K-1)$ .
6. процесс продолжается до тех пор, пока в каждой подгруппе не окажется по одному сообщению.

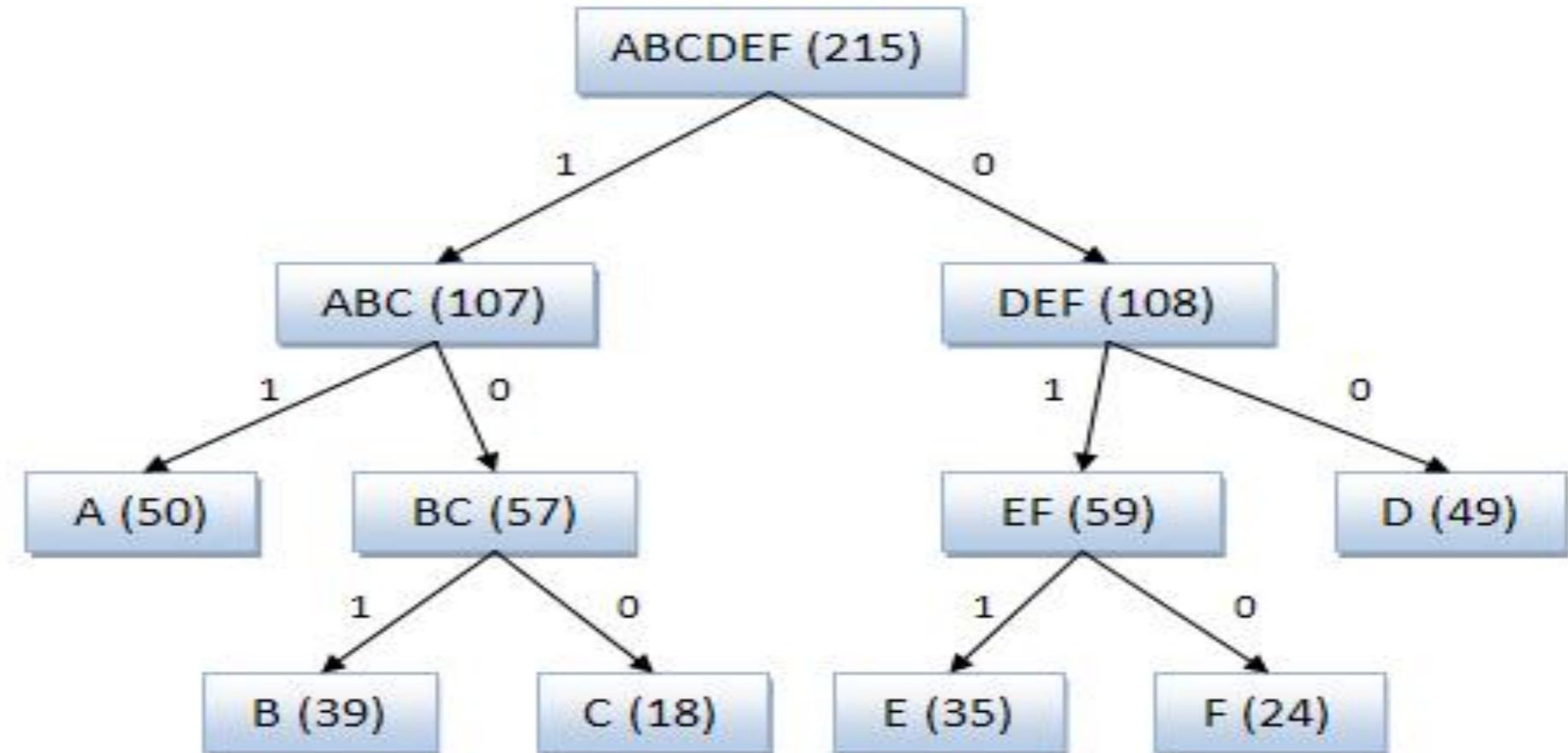
Символ	Вероятность, $P_i$
$A$	0,2
$E$	0,4
$F$	0,125
$C$	0,125
$B$	0,15



Символ	Вероятность, $P_i$
$E$	0,4
$A$	0,2
$B$	0,15
$F$	0,125
$C$	0,125



- A (частота встречаемости 50)
- B (частота встречаемости 39)
- C (частота встречаемости 18)
- D (частота встречаемости 49)
- E (частота встречаемости 35)
- F (частота встречаемости 24)



Полученный код: A — 11, B — 101, C — 100, D — 00, E — 011, F — 010.

# Метод Хаффмана

Классический алгоритм Хаффмана на входе получает таблицу частот встречаемости символов в сообщении. Далее на основании этой таблицы строится дерево кодирования Хаффмана (Н-дерево).

1. Символы входного алфавита образуют список свободных узлов. Каждый лист имеет вес, который может быть равен либо вероятности, либо количеству вхождений символа в сжимаемое сообщение.

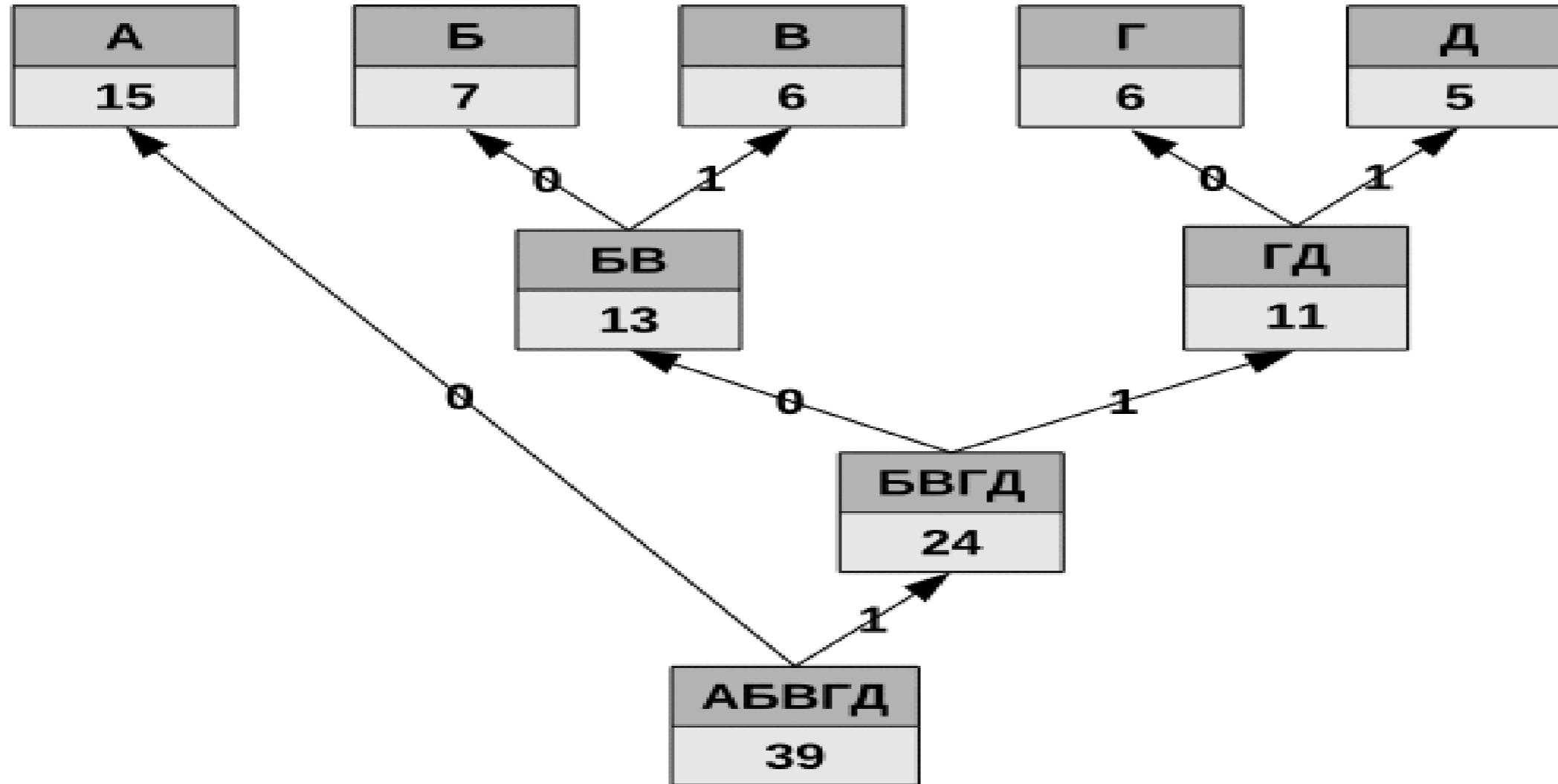
2. Выбираются два свободных узла дерева с наименьшими весами.

Создается их родитель с весом, равным их суммарному весу.

4. Родитель добавляется в список свободных узлов, а два его потомка удаляются из этого списка.

5. Одной дуге, выходящей из родителя, ставится в соответствие бит 1, другой – бит 0. Битовые значения ветвей, исходящих от корня, не зависят от весов потомков.

Шаги, начиная со второго, повторяются до тех пор, пока в списке свободных узлов не останется только один свободный узел. Он и будет считаться корнем дерева.



**Итого:**

<b>А</b>	<b>Б</b>	<b>В</b>	<b>Г</b>	<b>Д</b>
0	100	101	110	111

При **неравномерном** кодировании вводят среднюю длину кодировки, которая определяется по формуле

$$L = \sum_{i=1}^n P_i L_i$$

В общем же случае связь между средней длиной кодового слова  $L$  и энтропией  $H$  источника сообщений дает следующая **теорема кодирования** Шеннона:

имеет место неравенство  $L \geq H$ , причем  $L = H$  тогда, когда набор знаков можно разбить на точно равновероятные подмножества;

всякий источник сообщений можно закодировать так, что разность  $L - H$  будет как угодно мала.

Разность  $L - H$  называют **избыточностью кода** (мера бесполезно совершаемых альтернативных выборов).

следует не просто кодировать каждый знак в отдельности, а рассматривать вместо этого двоичные кодирования для  $nk$  групп по  $k$  знаков. Тогда средняя длина кода  $i$ -го знака  $x_i$  вычисляется так:

$$L = (\text{средняя длина всех кодовых групп, содержащих } x_i)/k.$$

Символ	Вероятность	Кодировка	Длина	$B \times D$
$A$	0,7	0	1	0,7
$B$	0,2	10	2	0,4
$C$	0,1	11	2	0,2

Средняя длина слова:  $L = 0,7+0,4+0,2=1,3.$

Среднее количество информации, содержащееся в знаке (энтропия):

$$H = 0,7 \times \text{Id}(1/0,7) + 0,2 \times \text{Id}(1/0,2) + 0,1 \times \text{Id}(1/0,1) = 0,7 \times 0,515 + 0,2 \times 2,322 + 0,1 \times 3,322 = 1,1571.$$

Избыточность  $L - H = 1,3 - 1,1571 = 0,1429.$

## Кодирование пар

Пары	Вероятность	Кодировка	Длина	$B \times D$
AA	0,49	0	1	0,49
AB	0,14	100	3	0,42
BA	0,14	101	3	0,42
AC	0,07	1100	4	0,28
CA	0,07	1101	4	0,28
BB	0,04	1110	4	0,16
BC	0,02	11110	5	0,10
CB	0,02	111110	6	0,12
CC	0,01	111111	6	0,06
Средняя длина кодовой группы из 2-х символов равна				2,33

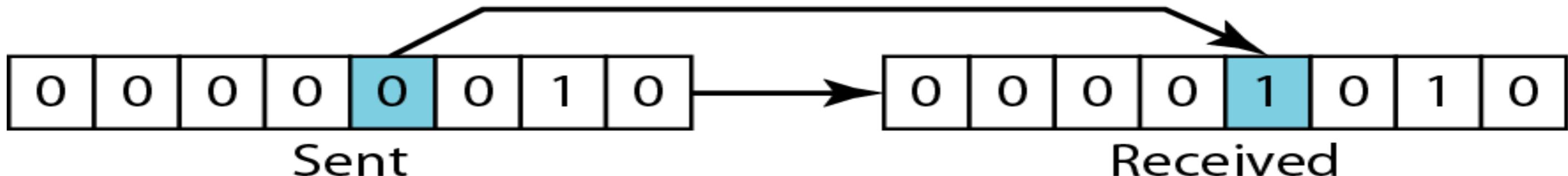
Средняя длина кода одного знака равна  $2,33/2=1,165$  – уже ближе к энтропии. Избыточность равна  $L - H = 1,165 - 1,1571 \approx 0,008$ .

# Помехоустойчивое кодирование

## Введение избыточности

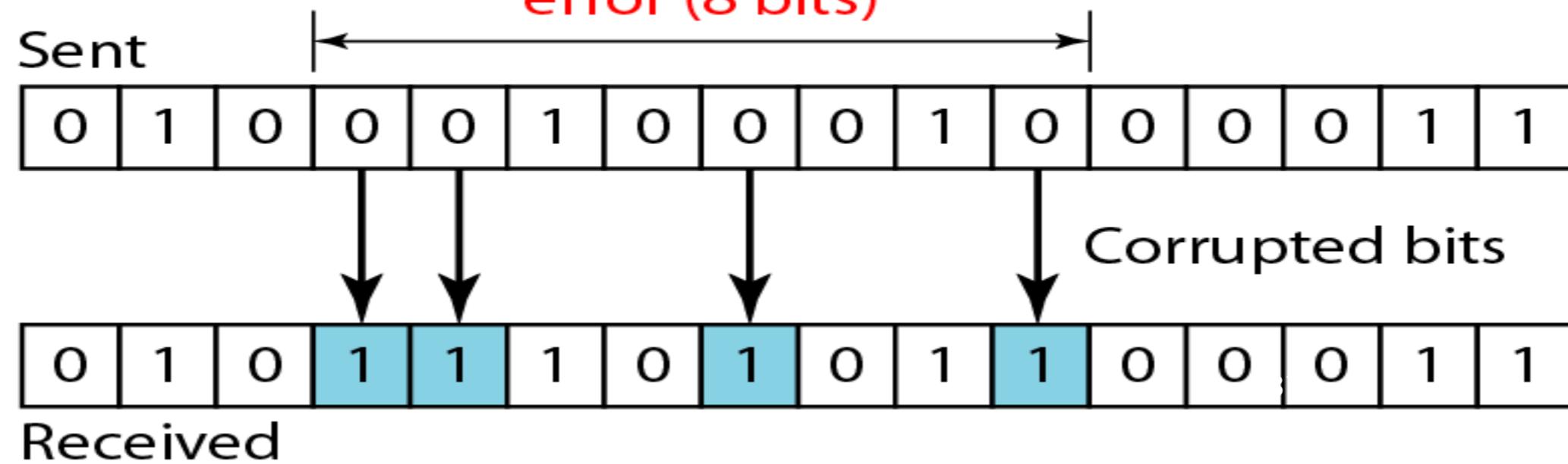
Ошибка в одном разряде

0 changed to 1



Пакет ошибок длины 8

Length of burst  
error (8 bits)



# Модель ошибки

*Ошибка – замена в двоичном сообщении 0 на 1 и\или наоборот, замена 1 на 0*

Пример: ИСХОДНОЕ СЛОВО: 00010100

Стирающий канал

1111101 ->111101

Канал со вставками

1111101->**0**1111101

ОШИБОЧНЫЕ СЛОВА: 00**1**010100, 000**0**0100,  
**00101**100

Расстояние Хэмминга между двумя словами есть число разрядов, в которых эти слова различаются

- 1. Расстояние Хэмминга  $d(000, 011)$  есть 2*
- 2. Расстояние Хэмминга  $d(10101, 11110)$  равно 3*

# Декодирование – исправление ошибки, если она произошла

Множество кодовых слов  $\{00000, 01101, 10110, 11011\}$

Если полученное слово  $10000$ , то декодируем в «ближайшее» слово  $00000$

Если полученное слово  $11000$  – то только обнаружение ошибки, так как два варианта:  $11000$  – в  $00000$  или  $11000$  – в  $11011$

# Самокорректирующиеся коды

Коды, в которых возможно автоматическое исправление ошибок, называются **самокорректирующимися**. Для построения самокорректирующегося кода, рассчитанного на исправление одиночных ошибок, одного контрольного разряда недостаточно.

количество контрольных разрядов  $k$  должно быть выбрано так, чтобы удовлетворялось неравенство:

$$2^k \geq k + m + 1$$

$$k \geq \log_2(k + m + 1)$$

где  $m$  — количество основных двоичных разрядов кодового слова

## Минимальные значения k при заданных значениях t

Диапазон t	k <sub>min</sub>
1	2
2-4	3
5-11	4
12-26	5
27-57	6

## 1. Блочные коды

Сообщение разбивается на блоки фиксированной длины, к каждому добавляются избыточные символы.

Примеры:

- **Коды Хэмминга** — исправляют одиночные ошибки;
- **Циклические коды (CRC)** — в основном для обнаружения ошибок;
- **Коды БЧХ и Рида–Соломона** — исправляют множественные ошибки, широко используются в хранении данных и связи.

## 2. Свёрточные коды

Кодирование зависит от текущих и предыдущих битов сообщения (с использованием сдвигового регистра).

Часто декодируются алгоритмом **Витерби**.

Применяются в спутниковой и мобильной связи.

**Дополнительно:**

- **Турбо-коды и LDPC-коды** — итеративные коды, близкие к пределу Шеннона; используются в 4G/5G, Wi-Fi, спутниковой связи.

**Ключевое различие:** блочные коды работают с фиксированными блоками, свёрточные — с потоком данных и памятью о прошлом.

# Циклические коды

Циклический код является линейным блочным кодом без памяти, кодер обрабатывает данные блоками одинакового размера, получая на основе какого-то определенного количества информационных бит, какое-то определенное число проверочных.  
Математически код основан на теории полей Галуа (операций по основанию остатка от деления на полином). Свойство циклического кода, в том, что каждая циклическая перестановка любого кодового слова порождает кодовое слово.

$$\begin{aligned}(c_0 + c_1x + c_2x^2 + \dots + c_{n-1}x^{n-1}) \cdot x = \\ (c_0x + c_1x^2 + c_2x^3 + \dots + c_{n-1}x^n) \bmod (x^n - 1)? \\ (x^n - 1) * c_{n-1} + z = c_0x + c_1x^2 + c_2x^3 + \dots + c_{n-1}x^n\end{aligned}$$

$$x^n c_{n-1} - c_{n-1} + z = c_0x + c_1x^2 + c_2x^3 + \dots + c_{n-1}x^n$$

Z ?

$$z = c_{n-1} + c_0x + c_1x^2 + c_2x^3 + \dots + c_{n-2}x^{n-1}$$

что соответствует циклическому сдвигу влево.

Например,  $19230 \bmod 9999 = 9231$

Код является циклическим тогда и только тогда, когда его порождающий многочлен  $g(X)$  **делит без остатка** многочлен  $X^n - 1$ .

# Кодирование и декодирование

1. Информационная последовательность представляется как многочлен  $m(X)$  степени  $k-1$ .
2.  $m(X)$  умножается на  $X^r$  (эквивалентно добавлению  $r$  нулей в конец).
3. Полученный многочлен  $X^r * m(X)$  **делится** на порождающий многочлен  $g(X)$ .
4. Остаток от деления  $r(X)$  (степени меньше  $r$ ) является проверочной (контрольной) частью.
5. **Кодовое слово** формируется как  $X^r * m(X) + r(X)$ . Оно гарантированно делится на  $g(X)$ .

## Декодирование (обнаружение ошибок):

1. Принятая последовательность (исходное слово + возможные ошибки) представляется как многочлен  $v(X)$ .
2. Многочлен  $v(X)$  **делится** на порождающий многочлен  $g(X)$ .

## 3. Анализ остатка:

- Если остаток равен нулю  $\rightarrow$  ошибок нет (или ошибка не обнаружима данным кодом).
- Если остаток не равен нулю  $\rightarrow$  при передаче произошла ошибка.

**Исправление ошибок** требует более сложных алгоритмов (например, алгоритм Маггитта, декодер Масси),

которые по виду остатка (синдрому) определяют место и характер ошибки.

# Сверточные коды

Сверточные коды получаются путем вставки между последовательностями информационных бит проверочных, при этом проверка потоковая, нет деления на блоки. Состояние сверточного кодера имеет память, то есть последующие биты зависят, не только от текущего кодируемого бита, но и от состояния кодера зависящего от предыдущих бит. Витерби показал, что наиболее оптимальными являются сверточные коды.

Например, рекуррентный код Финка получается следующим образом (строго нельзя отнести код Финка к чисто сверточным кодам, потому что в чем-то похож на блочный, но использует механизм сверточного порождения):

$$\begin{aligned} & A_1, B_1, A_2, B_2, A_3, B_3 \dots A_k, B_k | A_{k+1}, B_{k+1} \dots \\ & B_k = A_{i-s} \wedge A_{i+s+1} \text{ расчет проверочного символа } B. \end{aligned}$$

С учетом данного способа, ошибка исправляется если в проверочных битах  $k1=i-s-1$ , и  $k2=i+s$  ошибка, и  $k2-k1=2s+1$ . Например,  $s=0$ ,  $B1$  и  $B2$  неверные значит  $A2$  инвертировать. Сам проверочный символ  $k=i$  неверен если ошибка при  $A_{i-s}, A_{i+s+1}$

# Описание через порождающие полиномы

Часто сверточные коды описываются многочленами.

Допустим, зададим информационную последовательность:

$$A(x) = a_0 + a_1 X + a_2 X^2 + \dots \text{ а } \{0,1..\}$$

Формируемые передаваемые биты могут описываться следующим образом:

$$B_j(X) = b_0 + b_1 X + b_2 X^2 + \dots$$

$$B_j(X) = G_j(X)A(X)$$

где  $G$  – это порождающие полиномы.

При этом систематические коды не меняют информационную последовательность, а несистематические меняют.

# Пример сверточных кодов в виде структурной схемы

Пример сверточного кода Финка в терминах порождающих полиномов с  $s=1$ :

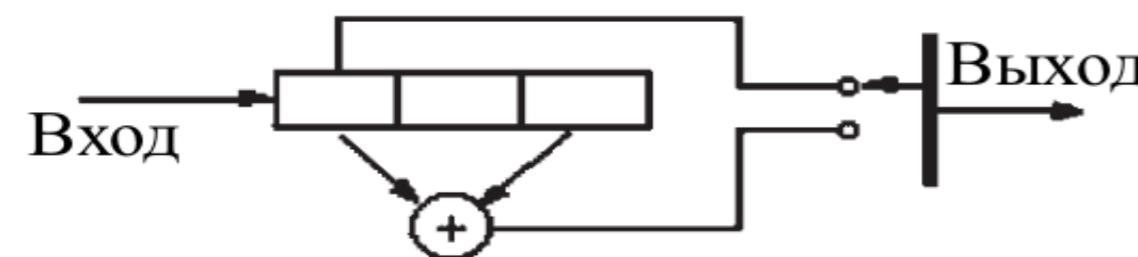
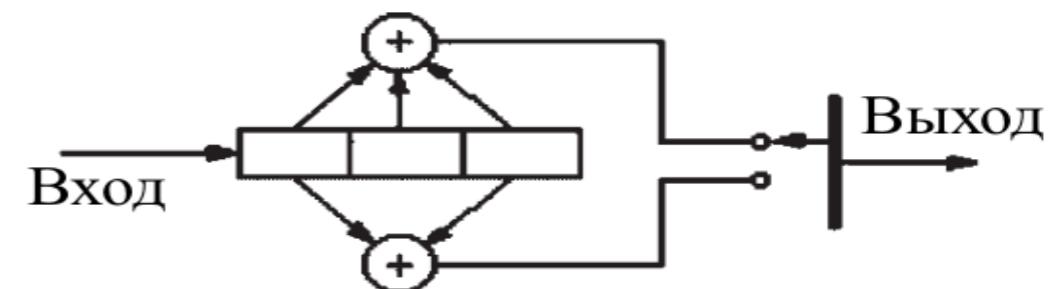


Схема порождающего полинома

$$G_1(X)=1; G_2(X)=1+X^2$$

Для кода Финка с  $s=0$  порождающие полиномы имеют вид:

$$G_1(X)=1; G_2(X)=1+X;$$



Пример несистематического сверточного кода

$$G_1(X)=1+X+X^2; G_2(X)=1+X^2$$

# Код Финка

$S = 0$

Маркировка символов, текущее значение $i$	$a_1 b_1$ 1	$a_2 b_2$ 2	$a_3 b_3$ 3	$a_4 b_4$ 4	$a_5 b_5$ 5	$a_6 b_6$ 6	$a_7 b_7$ 7	$a_8 b_8$ 8	$a_9 b_9$ 9	$a_{10} b_{10}$ 10
--	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	-----------------------

Информационные символы, суммирование по mod2, проверочные символы	0 0 0 0 0 1 1 0 1 1 0 1 1 1 0
---	---

Выходная последовательность	0 0 0 0 0 1 1 0 1 1 0 1 1 1 0
-----------------------------	---

## Декодирование при ошибке в одном информационном символе

Принятая последовательность, суммирование по mod 2, результат суммирования	0 0 0 0 0 1 1 0 0 1 1 0 1 1 0 1
Декодированная последовательность	0 0 0 0 0 1 1 0 1 1 0 1 1 1 0

## Декодирование при ошибке в одном проверочном символе

Принятая последовательность, суммирование по mod 2, результат суммирования	0 0 0 0 0 1 1 0 0 1 1 0 1 1 0 1
Декодированная последовательность	0 0 0 0 0 1 1 0 1 1 0 1 1 1 0

## Декодирование при ошибке в двух проверочных символах

Принятая последовательность, суммирование по mod 2, результат суммирования	0 0 0 0 0 1 0 0 1 1 1 0 1 1 0 1
Декодированная последовательность	0 0 0 0 0 1 1 0 1 1 0 1 1 1 0

→ |  $l_c = 3$  | ←

# Код Хэмминга, восстановление одного искажения или обнаружение двойного, неклассический подход

110 | 101 | 100 | 011 | 010 | 001

6 5 4 3 2 1

101011

001

001  
010  
100  
110

XOR

00 0  
01 1  
10 1  
11 0

100011

001

001  
010  
110

101

101

001

100

4

Для Примера рассмотрим классический код Хемминга  
Сгруппируем проверочные символы следующим образом:

$$r_1 = i_1 \oplus i_2 \oplus i_3$$

$$r_2 = i_2 \oplus i_3 \oplus i_4$$

$$r_3 = i_1 \oplus i_2 \oplus i_4$$

Получение кодового слова выглядит следующим образом:

$$(i_1 \quad i_2 \quad i_3 \quad i_4) \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix} = (i_1 \quad i_2 \quad i_3 \quad i_4 \quad r_1 \quad r_2 \quad r_3)$$

# Декодирование

На вход декодера поступает кодовое слово

$$V = (i'_1, i'_2, i'_3, i'_4, r'_1, r'_2, r'_3)$$

В декодере в режиме исправления ошибок строится последовательность синдромов:

$$S_1 = r_1 \oplus i_1 \oplus i_2 \oplus i_3$$

$$S_2 = r_2 \oplus i_2 \oplus i_3 \oplus i_4$$

$$S_3 = r_3 \oplus i_1 \oplus i_2 \oplus i_4$$

$$\sim i_2 + i_2 = 1$$

Получение синдрома выглядит следующим образом:

$$(i_1 \ i_2 \ i_3 \ i_4 \ r_1 \ r_2 \ r_3) \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = (S_1 \ S_2 \ S_3)$$

Нули в синдроме показывают отсутствие ошибок, отличный от нуля код соответствует какой-либо единичной ошибке, например для 111, это ошибка в  $i_2$

# Получение кода хэмминга для

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	
	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	1	1	1	0	1	
1	x		x		x		x		x		x		x		x		x		x		x	1
0		x	x			x	x			x	x			x	x			x	x		2	
1			x	x	x	x					x	x	x	x					x	x	4	
0							x	x	x	x	x	x	x	x							8	
0																x	x	x	x	x	16	

Каждую последовательность суммируем по модулю 2 (операция xor), получая код:

$$0+0+1+0+0+0+1+1+1+1=1$$

$$0+0+0+0+1+0+0+1+1+1=0$$

$$0+1+0+0+0+0+0+1+0+1=1$$

$$0+0+1+0+0+0+0+0+1=0$$

$$0+1+1+1+0+1=0$$

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	
	1	0	0	1	1	0	0	0	0	1	0	0	0	0	1	0	1	1	1	0	1	
1	0		0	1	1	0	0	0	0	1	0	0	0	0	1	0	1	1	1	0	1	
0		x	x			x	x			x	x			x	x			x	x		2	
1			x	x	x	x					x	x	x	x					x	x	4	
0							x	x	x	x	x	x	x	x							8	
0																x	x	x	x	x	16	

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	
1	0	0	1	1	0	0	0	0	1	0	0	0	0	1	0	1	1	1	0	1	
X		X		X		X		X		X		X		X		X		X		X	1
X	X			X	X			X	X			X	X			X	X				2
		X	X	X	X					X	X	X	X					X	X		4
						X	X	X	X	X	X	X	X								8
															X	X	X	X	X	X	16

$$1+0+1+0+0+1+0+1+1+1 = 11$$

$$0+0+0+0+1+1+0+1+1+1 = 12$$

$$1+1+0+0+0+0+0+1+0+1 = 04$$

$$0+0+1+1+0+0+0+1 = 18$$

$$0+1+1+1+0+1 = 016$$

11

# Понятие системы счисления

**Система счисления** — это способ записи чисел с помощью заданного набора специальных знаков.

# Два вида систем счисления

## позиционные

В позиционных системах счисления вес каждой цифры изменяется в зависимости от ее **положения** (позиции) в последовательности цифр, изображающих число

Пример: В десятичном числе 757,7 первая семерка означает 7 сотен, вторая – 7 единиц, третья – 7 десятых долей единицы.

## непозиционные

В непозиционных системах вес цифры (т.е. тот вклад, который она вносит в значение числа) не зависит от ее позиции в записи числа.

Пример: В римской системе счисления в числе XXXII (тридцать два) вес цифры X в любой позиции равен просто десяти.

Степень двойки	Десятичное	Восьмеричное
$2^0$	1	1
$2^1$	2	2
$2^2$	4	4
$2^3$	8	10
$2^4$	16	20
$2^5$	32	40
$2^6$	64	100
$2^7$	128	200
$2^8$	256	400
$2^9$	512	1000
$2^{10}$	1024	2000
$2^{11}$	2048	4000
$2^{12}$	4096	10000

при работе с двоичными кодами удобны недесятичные системы счисления, а основания кратные степеням двойки.

**Любая позиционная система счисления характеризуется своим основанием**

Любое двоичное число, состоящее из 1 с несколькими нулями, является степенью двойки. Показатель степени равен числу нулей.

Таблица степеней двойки демонстрирует это правило наглядно.

Степень двойки	Десятичное	Восьмеричное	Четверичное	Двоичное
$2^0$	1	1	1	1
$2^1$	2	2	2	10
$2^2$	4	4	10	100
$2^3$	8	10	20	1000
$2^4$	16	20	100	10000
$2^5$	32	40	200	100000
$2^6$	64	100	1000	1000000
$2^7$	128	200	2000	10000000
$2^8$	256	400	10000	100000000
$2^9$	512	1000	20000	1000000000
$2^{10}$	1024	2000	100000	10000000000

Переводимое число необходимо записать в виде суммы произведений цифр числа на основание системы счисления в степени, соответствующей позиции цифры в числе.

5 4 3 2 1 0 -1 -2

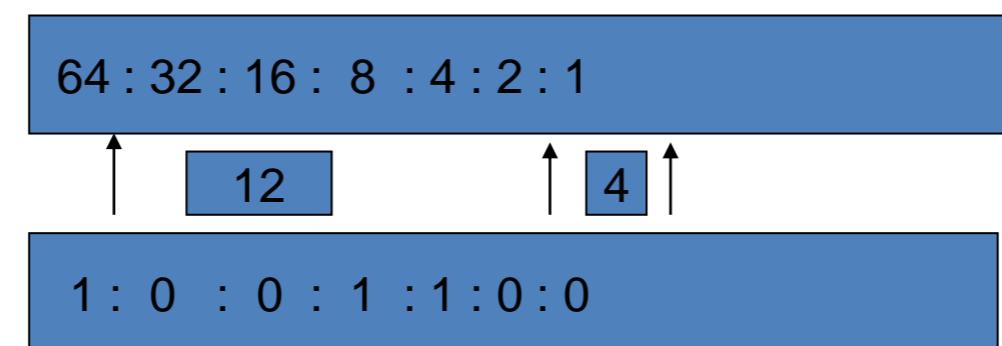
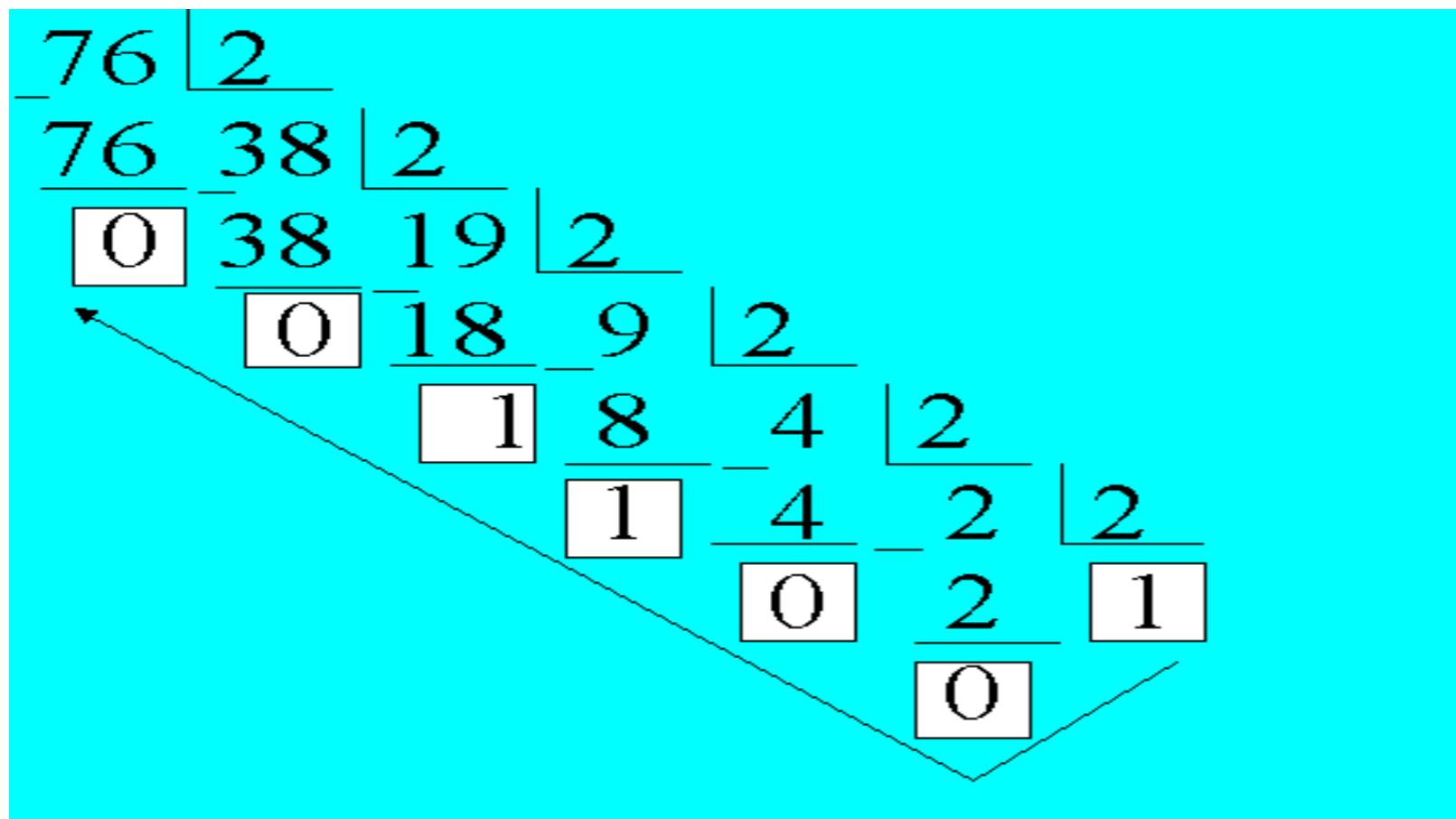
$$111000.11_2 =$$

$$1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^{-1} + 1 \cdot 2^{-2} =$$

$$= 32 + 16 + 8 + \frac{1}{2} + \frac{1}{4} =$$

$$= 56,75_{10}$$

$1001100_2$



$$\begin{array}{r} \times 0,375 \\ \times 2 \\ \hline 0,750 \\ \times 2 \\ \hline 1,500 \\ \times 2 \\ \hline 1,000 \end{array}$$

$$0,375_{10} =$$

$$0,011_2$$

Дробная часть получается из целых частей (0 или 1) при ее последовательном умножении на 2 до тех пор, пока дробная часть не обратится в 0 или получится требуемое количество знаков после разделительной точки.

# Пример перевода из восьмиричной системы

2 1 0 -1

$$\begin{aligned}421.5_8 &= 4 \cdot 8^2 + 2 \cdot 8^1 + 1 \cdot 8^0 + 5 \cdot 8^{-1} = \\&= 256 + 16 + 1 + 5/8 = \\&= 273,625_{10}\end{aligned}$$

# Пример перевода из шестнадцатиричной системы

1 0 -1

$$\begin{aligned} A7.C_{16} &= 10 \cdot 16^1 + 7 \cdot 16^0 + 12 \cdot 16^{-1} = \\ &= 160 + 7 + 12/16 = \\ &= 167,75_{10} \end{aligned}$$

# Запись в десятичной, двоичной, восьмеричной и шестнадцатеричной системах счисления первых двух десятков целых чисел

10 - я	2 - я	8 - я	16 - я
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9

10 - я	2 - я	8 - я	16 - я
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10
17	10001	21	11
18	10010	22	12
19	10011	23	13

# Примеры перевода из двоичной системы счисления в восьмеричную

$100110111.001_2 =$  100 110 111. 001<sub>2</sub>

$100110111.001_2 =$  4 6 7. 1<sub>8</sub>

$10100101110.11_2 =$  010 100 101 110. 110<sub>2</sub>

$10100101110.11_2 =$  2 4 5 6. 6<sub>8</sub>

# Перевод из восьмеричной системы счисления в двоичную

Такой перевод осуществляется путем подстановки: каждая 8-ричная цифра заменяется на соответствующие ей три двоичных.

$$74.6_8 = \begin{array}{c} 111 \\ 100. \\ 110_2 \end{array}$$

$$310.5_8 = \begin{array}{c} 011 \\ 001 \\ 000. \\ 101_2 \end{array}$$

## Кодировки символов

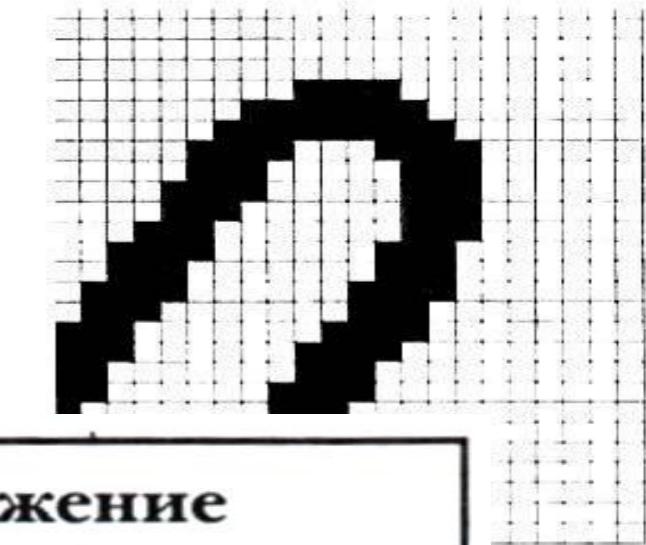
Двоичный код	Десятичный код	КОИС	CP1251	CP866	Mac	ISO
00000000	0				,	
.....						
00001000	8				Удаление последнего символа (клавиша Backspace)	
.....						
00001101	13			Перевод строки (клавиша Enter)		
.....						
00100000	32			Пробел		
00100001	33			!		
.....						
01011010	90				z	
.....						
01111111	127			□		
10000000	128	-	ъ	А	А	к
.....						
11000010	194	б	в	-	-	т
.....						
11001100	204	л	м	l	j	ъ
.....						
11011101	221	щ	э	-	€	н
.....						
11111111	255	ъ	я	Нераздел. пробел	Нераздел. пробел	п

# Двоичное кодирование графической информации

В простейшем случае (черно-белое изображение без градаций серого цвета). Каждая точка экрана может иметь лишь два состояния – «черная» или «белая», т.е. для хранения ее состояния необходим 1 бит.

**Восьмицветная палитра**  
(на основе базовых цветов)

R	G	B	Цвет
0	0	0	черный
0	0	1	синий
0	1	0	зеленый
0	1	1	голубой
1	0	0	красный
1	0	1	розовый
1	1	0	коричневый
1	1	1	белый



**Монохромное изображение**  
(черно-белый монитор)

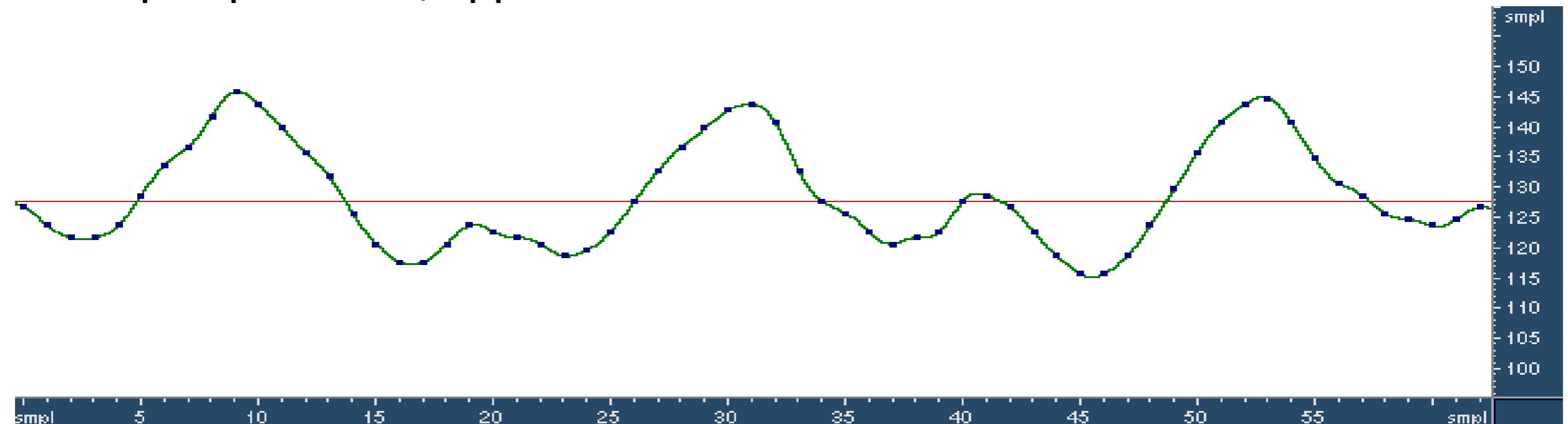
	0	1 бит видеопамяти
	1	

# Представление цветов rgb

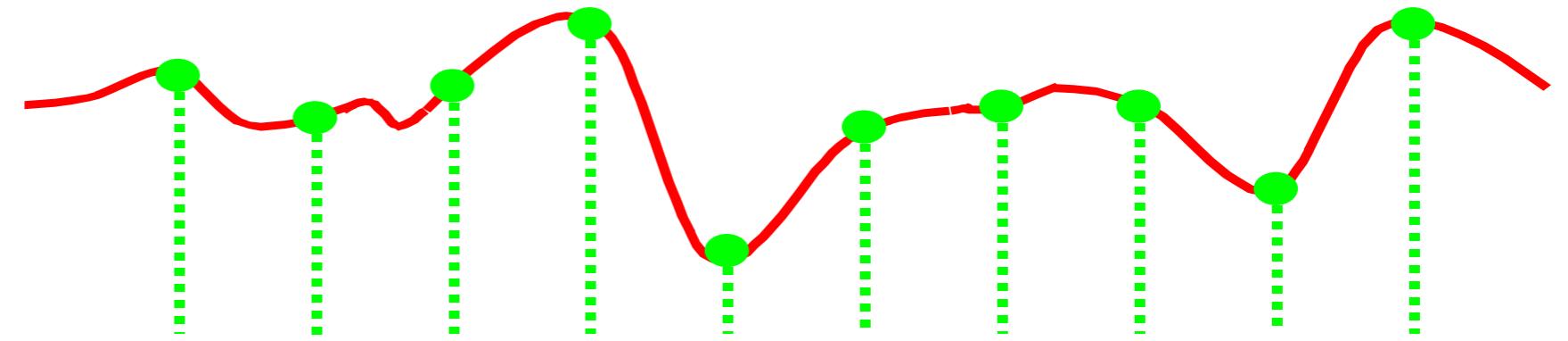
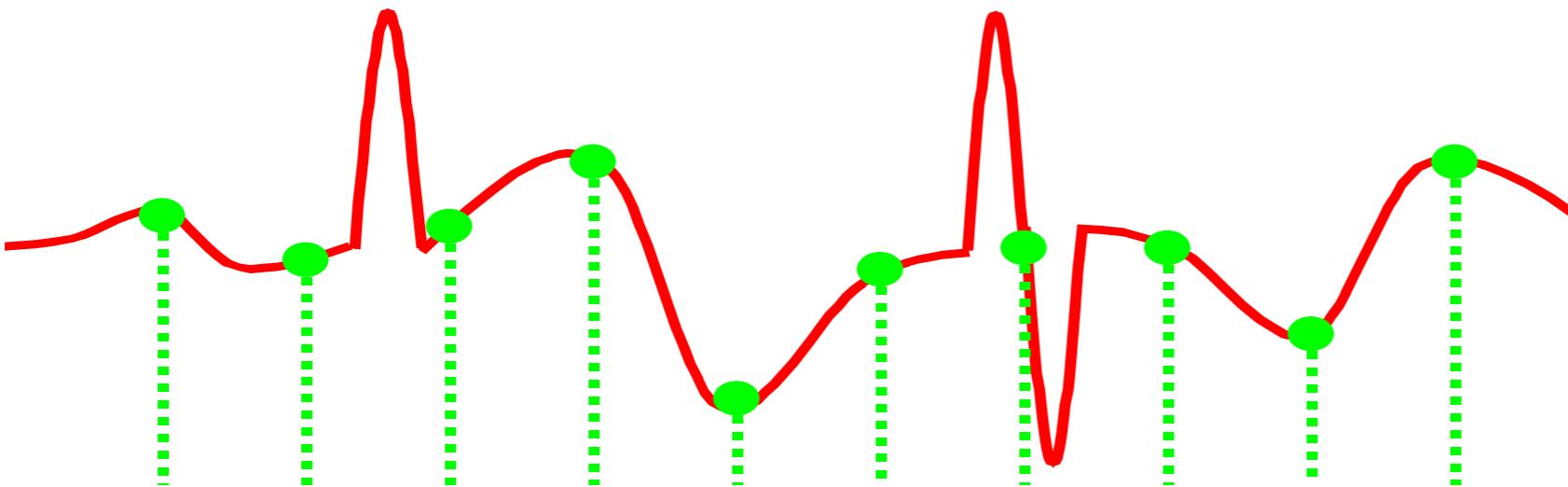
R	G	B	Цвет
255	0	0	Красный
0	0	255	Синий
0	255	0	Зеленый
0	0	0	Черный
255	255	255	Белый

# Мультимедийная информация

- Звук
  - Запись и оцифровка
  - Частота и разрядность дискретизации
  - Артефакты оцифровки



## Точечная выборка



- часть информации потеряна!

Определение: Преобразование чисел высокой точности в числа низкой точности

Зачем?

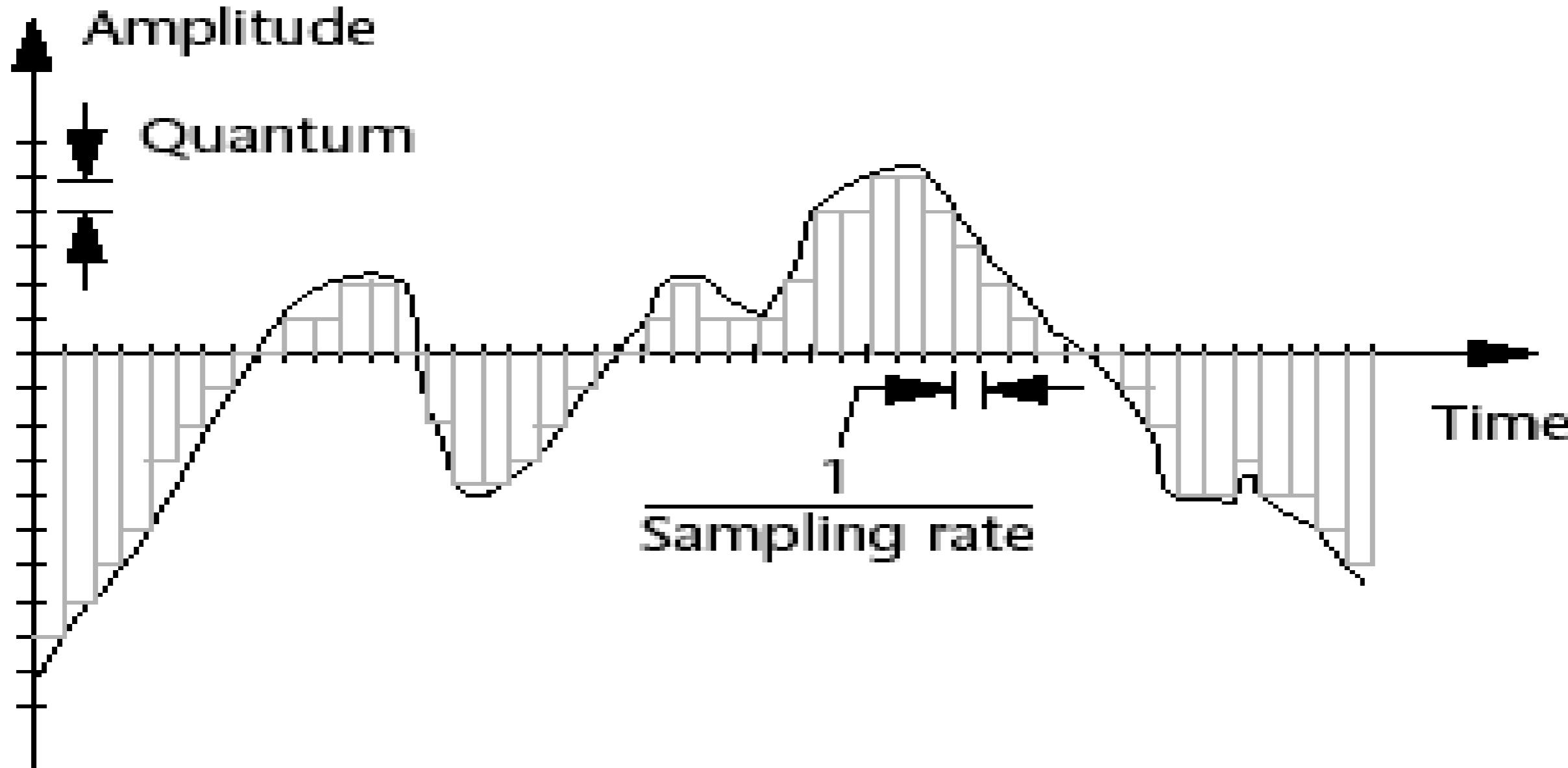
Экономия памяти

Вывод на двоичные устройства

Как?

Минимизация ошибки (скорее, ошибки восприятия)

# Дискретизация и квантование аналоговой волны



## Скорость передачи

Пример

256 уровней квантования

Значит для кодирования надо 8 бит

Частота дискретизации 8000 Гц, значит 8000 раз в секунду делаются отчеты

Скорость передачи –  $8000 * 8 = 64$  кбит/с

Количество бит \* Частоту дискретизации [бит/с]

## Дополнительный код

Число полученное путем вычитания из числа с числом разрядов больше на 1, и со значением 1 в старшем разряде и 0 младших. Пример 1000. Для числа 70, дополнительный код  $100-70=30$  наиболее распространённый способ представления отрицательных целых чисел в компьютерах. Он позволяет заменить операцию вычитания на операцию сложения и сделать операции сложения и вычитания одинаковыми для знаковых и беззнаковых чисел, чем упрощает архитектуру ЭВМ.

$$59-41 = ? \quad 18$$

Доп код 41,  $100-41 = 59$

Можно представить как:

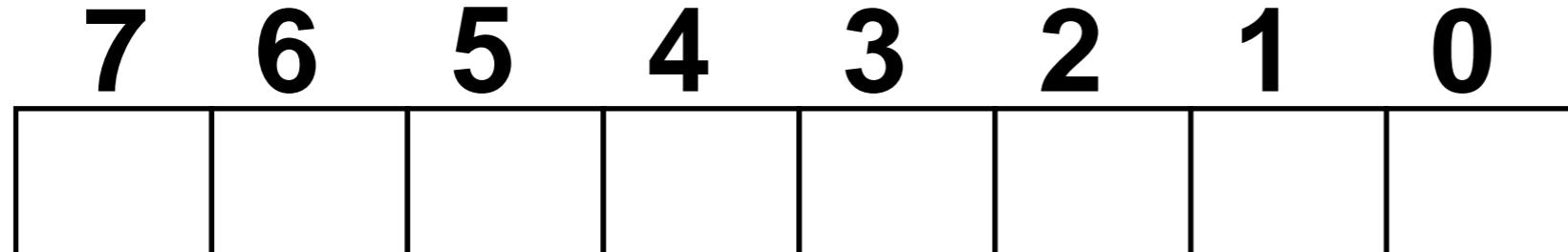
$$59-(100-59) = 59+59 - 100 = \cancel{1}18-\cancel{1}00 = 18$$

- В двоичной системе
- Доп кол получается как
- $10000-1001\dots$
- Что такое  $10000$ , это  $1111+1$
- $1111-1001$  получается путем инвертирования  $0110$
- Остается добавить 1, чтобы получить доп код
- $0111$

## Пример 1

Представить число  $21_{10}$  и  $-21_{10}$  в однобайтовой разрядной сетке.

1. Переведем число  $21_{10}$  в двоичную систему счисления.  $21_{10} = 10101_2$ .
2. Нарисуем восьмиразрядную сетку (1 байт = 8 бит).



## Пример 1

---

**3.** Впишем число, начиная с младшего разряда.

			1	0	1	0	1
--	--	--	---	---	---	---	---

**4.** Заполним оставшиеся разряды нулями.

0	0	0	1	0	1	0	1
---	---	---	---	---	---	---	---

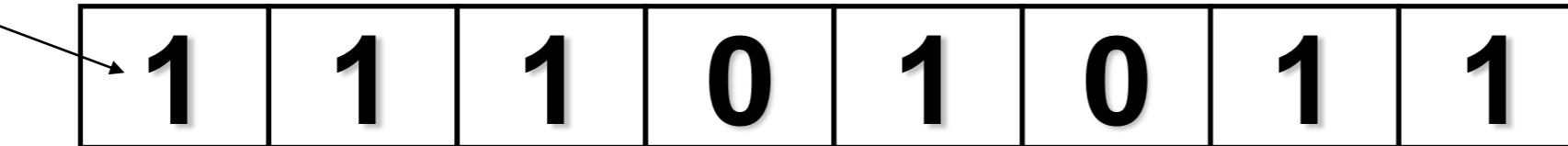
## Пример 1

### 5. Инвертируем

1	1	1	0	1	0	1	0
---	---	---	---	---	---	---	---

### 6. Прибавляем 1

знак



1	1	1	0	1	0	1	1
---	---	---	---	---	---	---	---

Проверка

0	0	0	1	0	1	0	0
---	---	---	---	---	---	---	---

0	0	0	1	0	1	0	1
---	---	---	---	---	---	---	---

→ 21

Расширение числа, например, от байта до слова (два байта) или до двойного слова (четыре байта) делается путем добавления единиц слева, если это число отрицательное в дополнительном коде и нулей если число в прямом коде

1	1	1	1	1	1	1	1
---	---	---	---	---	---	---	---

1	1	1	0	1	0	1	1
---	---	---	---	---	---	---	---

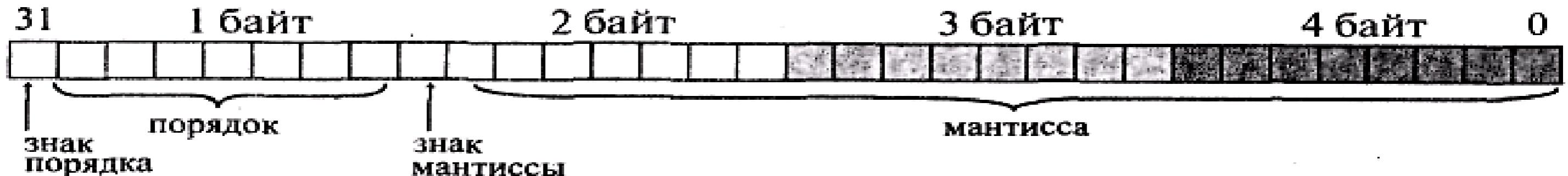
0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---

0	0	0	1	0	1	0	1
---	---	---	---	---	---	---	---

# Кодирование вещественных чисел

Число в форме с плавающей точкой занимает в памяти компьютера **четыре** (число обычной точности) байта или **восемь** (число двойной точности) байта.

Для записи чисел в разрядной сетке выделяются разряды для знака порядка и мантиссы, для порядка и для мантиссы.



## Пример 3

---

Представить число 250,187510 в формате с плавающей точкой в 4-байтовой разрядной сетке:

**1.** Переведем число в двоичную систему счисления с 23 значащими цифрами:

$$250,1875_{10} = 11111010,001100000000000_2;$$

**2.** Нормализуем мантиссу:

$$11111010,001100000000000 = 0,1111101000110000000 \cdot 10^{1000};$$

## Пример 3

3.  $0,1111101000110000000000 \cdot 10^{1000}$ ;

(мантиssa положительная)

(порядок положительный)

4. Запишем число в 32-разрядной сетке:

