# Глава 1
# Информация и информационные процессы

## § 1
## Количество информации

**Формула Хартли**

Вы знаете, что при выборе из двýх возможных вариантов количество полученной информации равно 1 биту. Если количество вариантов $N$ равно $2^I$, то количество информации при выборе одного из них равно $I$ битов. А как вычислить количество информации, если количество вариантов не равно степени числа 2?

Ответить на этот вопрос стало возможно только после того, как вы изучили логарифмы в курсе математики. Из формулы

$$N = 2^I$$

сразу следует, что $I$ — это степень, в которую нужно возвести 2, чтобы получить $N$, т. е. *логарифм*:

$$I = \log_2 N.$$

Эта формула называется **формулой Хартли** в честь американского инженера Ральфа Хартли, который предложил её в 1928 г.

**Ральф Хартли (1888–1979)**

Пусть, например, на лётном поле стоят 10 самолётов (с номерами от 1 до 10) и известно, что один из них летит в Санкт-Петербург. Сколько информации в сообщении «Самолёт № 2 летит в Санкт-Петербург»? У нас есть 10 вариантов, из которых выбирается один, поэтому по формуле Хартли количество информации равно

$$I = \log_2 10 \approx 3{,}322 \text{ бита}.$$

Обратите внимание, что для значений $N$, которые не равны целой степени числа 2, количество информации в битах — дробное число.

С помощью формулы Хартли можно вычислить теоретическое количество информации в сообщении. Предположим, что алфавит (полный набор допустимых символов) включает 50 символов (в этом случае говорят, что **мощность алфавита** равна 50). Тогда информация при получении каждого символа составляет

$$I = \log_2 50 \approx 5{,}644 \text{ бита}.$$

Если сообщение содержит 100 символов, его общий информационный объём примерно равен

$$5{,}644 \cdot 100 = 564{,}4 \text{ бита}.$$

В общем случае объём сообщения длиной L символов, использующего алфавит из N символов, равен $I = L \cdot \log_2 N$.

Такой **подход** к определению количества информации называют **алфавитным**. Конечно, на практике невозможно использовать для кодирования символа нецелое число битов, поэтому используют первое целое число, которое больше теоретически рассчитанного значения. Например, при использовании алфавита из 50 символов каждый символ будет закодирован с помощью 6 битов ($50 \le 2^6 = 64$).

Сколько разных сообщений можно передать, если известен алфавит и длина сообщения? Предположим, что для кодирования сообщения используются 4 буквы, например «А», «Б», «В» и «Г», и сообщение состоит из двух символов. Поскольку каждый символ может быть выбран 4 разными способами, на каждый вариант выбора первого символа есть 4 варианта выбора второго. Поэтому общее число разных двухбуквенных сообщений вычисляется как $4 \cdot 4 = 4^2 = 16$. Если в сообщение добавить ещё один символ, то для каждой из 16 комбинаций первых двух символов третий можно выбрать четырьмя способами, так что число разных трёхсимвольных сообщений равно $4 \cdot 4 \cdot 4 = 4^3 = 64$.

В общем случае, если используется алфавит из N символов, то количество разных возможных сообщений длиной L символов равно $Q = N^L$.

## Задачи

1. В русском лото 99 бочонков. Какое количество информации содержится в сообщении «Первым вытащили бочонок с номером 16»?

2. В классе 25 учеников. Какое количество информации содержится в сообщении «Сегодня дежурит Василий Иванов»?
3. В чём состоит алфавитный подход к оценке количества информации?
4. Оцените теоретическое количество информации в сообщении «Приеду в четверг». Используемый алфавит состоит из заглавных и строчных русских букв и пробела.
5. Каков будет фактический объём сообщения «Приеду в четверг», если при передаче каждый символ кодируется минимально возможным целым числом битов?
6. В некоторой стране автомобильный номер длиной 7 символов составляется из заглавных букв (всего используется 26 букв) и десятичных цифр в любом порядке. Каждый символ кодируется одинаковым и минимально возможным количеством битов, а каждый номер — одинаковым и минимально возможным количеством байтов. Определите объём памяти, необходимый для хранения 20 автомобильных номеров.
7. Объём сообщения, содержащего 4096 символов, равен 1/512 части мегабайта. Какова мощность алфавита, с помощью которого записано это сообщение?
8. Какое наименьшее число символов должно быть в алфавите, чтобы с помощью всевозможных трёхбуквенных слов, состоящих из символов данного алфавита, можно было передать не менее 9 различных сообщений?

*Таблица 1.1*

| | |
|---|---|
| **Пробел** | 0,175 |
| **О** | 0,090 |
| **Е, Ё** | 0,072 |
| **А** | 0,062 |
| **И** | 0,062 |
| **Т** | 0,053 |
| **Н** | 0,053 |
| **С** | 0,045 |
| **Р** | 0,040 |
| **В** | 0,038 |
| **Л** | 0,035 |
| **К** | 0,028 |
| **М** | 0,026 |
| **Д** | 0,025 |
| **П** | 0,023 |
| **У** | 0,021 |
| **Я** | 0,018 |
| **Ы** | 0,016 |
| **З** | 0,016 |
| **Ь, Ъ** | 0,014 |
| **Б** | 0,014 |
| **Г** | 0,013 |
| **Ч** | 0,012 |
| **Й** | 0,010 |
| **Х** | 0,009 |
| **Ж** | 0,007 |
| **Ю** | 0,006 |
| **Ш** | 0,005 |
| **Ц** | 0,004 |
| **Щ** | 0,003 |
| **Э** | 0,003 |
| **Ф** | 0,002 |

### Информация и вероятность

Всё, что написано в предыдущем пункте, верно только при одном уточнении: все события (символы алфавита) одинаково ожидаемы, т. е. нельзя заранее сказать, что какой-то символ встречается чаще, а какой-то — реже. В реальности это предположение не всегда верно. Например, в тексте на русском языке некоторые символы встречаются часто, а некоторые — очень редко. Числа во втором столбце табл. 1.1 означают относительную долю символа в больших текстах. Например, доля 0,175 для пробела означает, что примерно 17,5% всех символов в текстах на русском языке — пробелы.

Иногда среди всех возможных событий есть ожидаемые и неожиданные. Например, на вопрос «Идёт ли сейчас снег?» летом мы ожидаем услышать ответ «Нет». При этом ответ «Да» будет очень неожиданным, и после его получения наши дальнейшие планы могут сильно измениться. Это значит, что при таком ответе мы получаем гораздо больше информации. Как её измерить точно?

Сначала нужно разобраться с тем, что значит «менее ожидаемое» событие и «более ожидаемое». Математики в этом случае используют понятие **«вероятность»**: если событие более ожидаемое, то его вероятность (точнее, вероятность того, что оно произойдёт) больше.

Вероятность — это число в интервале от 0 до 1. В математике вероятность принято обозначать буквой $p$ (от латинского *probabilis* — вероятный, возможный).

Сначала рассмотрим предельные случаи, когда вероятность равна 0 или 1. Пусть, например, $x$ — некоторое неизвестное вещественное число, которое задумал ведущий. Вы знаете, что для любого вещественного $x$ *всегда* $x^2 \geq 0$. В этом случае считают, что вероятность события $x^2 \geq 0$ равна 1 (событие $x^2 \geq 0$ обязательно произойдёт). Часто вероятность измеряют в процентах от 1, тогда вероятность события $x^2 \geq 0$ равна 100%. В то же время вероятность события $x^2 < 0$ равна нулю, это значит, что событие $x^2 < 0$ никогда не произойдёт.

Теперь предположим, что мы бросаем монету и смотрим, какой стороной она упала, «орлом» или «решкой». Если повторять этот опыт много раз, мы заметим, что количество «орлов» и «решек» примерно равно (конечно, если монета не имеет дефектов). При этом вероятность каждого из двух событий равна 0,5, или 50%. Скорее всего, вы слышали выражение «50 на 50», которое означает, что ни одному из двух вариантов нельзя отдать предпочтение — их вероятности равны.

Вероятность события можно определить с помощью большого количества испытаний. Если из $N$ испытаний нужное нам событие случилось $m$ раз, то вероятность такого события можно оценить как $\frac{m}{N}$.

Например, классический игральный кубик имеет 6 граней; если кубик качественный, вероятность выпадения каждой грани равна 1/6. Вероятность выпадения чётного числа можно подсчитать так: среди чисел от 1 до 6 всего

3 чётных числа, поэтому при большом числе испытаний в половине случаев (в 3 из 6) будут выпадать чётные числа, т. е. вероятность равна 0,5. А вероятность выпадения числа, меньшего 3, равна $2/6 = 1/3 \approx 0{,}33$, потому что только 2 из 6 чисел (1 и 2) удовлетворяют условию.

Теперь можно переходить к главному вопросу: как вычислить количество информации, если в сообщении получен символ, вероятность появления которого равна $p$. Попробуем сначала определить, какими свойствами должна обладать эта величина, исходя из «здравого смысла».

Во-первых, чем меньше вероятность, тем более неожидан символ и тем больше информации мы получили. Если вероятность события близка к нулю, количество информации должно стремиться к бесконечности (получение такого символа очень неожиданно).

Во-вторых, представим себе, что мы получаем символы только одного вида, например только буквы «А». Тогда вероятность появления символа «А» равна 1 и никакой новой информации в этом символе для нас нет — мы всё заранее знали. Следовательно, при $p = 1$ информация должна быть равна нулю.

В-третьих, «здравый смысл» подсказывает, что когда мы бросаем игральный кубик два (три, 102, 1002) раза, мы получаем информации в два (три, 102, 1002) раза больше, чем при однократном бросании кубика. Это свойство называют принципом *аддитивности* (сложения).

Математики доказали, что этими свойствами обладает только логарифмическая функция вида $f(p) = -K \cdot \log_2 p$, где коэффициент $K$ можно выбирать произвольно, как удобно в конкретной задаче. Если взять $K = 1$, мы получим количество информации в битах.

---

Если событие имеет вероятность $p$, то количество информации в битах, полученное в сообщении об этом событии, равно

$$I = -\log_2 p = \log_2 \frac{1}{p}. \qquad (1)$$

---

Поскольку вероятность $p$ не больше 1, количество информации не может быть меньше нуля. Легко проверить, что если вероятность равна 1, то количество полученной информации равно нулю. Если вероятность стремится к 0, величина $\log_2 p$ стремится к $-\infty$, а количество информации — к $\infty$.

Третье свойство (аддитивность, сложение вероятностей) проверим на примере. Пусть есть два мешка, в каждом из которых лежат 8 шариков разного цвета. Вычислим, какое количество информации мы получили из сообщения «Из первого мешка вытащили (наугад) красный шарик, а из второго — зелёный». Из каждого мешка можно вытащить один из восьми шариков, т. е. вероятность вытащить какой-то определённый шарик равна 1/8. Следовательно, количество информации в сообщении «Из первого мешка вытащили красный шарик» равно

$$I = \log_2 \frac{1}{\frac{1}{8}} = \log_2 8 = 3 \text{ бита.}$$

Точно такую же информацию, 3 бита, мы получаем из сообщения «Из второго мешка вытащили зелёный шарик».

Теперь посмотрим, какое количество информации несёт исходное полное сообщение «Из первого мешка вытащили (наугад) красный шарик, а из второго — зелёный». Какова вероятность именно этого исхода? Есть 8 способов вытащить шарик из каждого мешка, всего получаем 8·8 = 64 варианта, поэтому вероятность каждого из них равна 1/64. Тогда количество информации в сообщении равно

$$I = \log_2 \frac{1}{\frac{1}{64}} = \log_2 64 = 6 \text{ битов,}$$

т. е. равно сумме количеств информации в двух отдельных сообщениях. Мы показали, что свойство аддитивности выполняется.

Возможно, вы уже заметили, что последний пример фактически свёлся к использованию формулы Хартли. Если все $N$ вариантов имеют равные вероятности, то вероятность каждого варианта равна $p = \frac{1}{N}$, следовательно, количество информации о любом из возможных событий вычисляется как

$$I = \log_2 \frac{1}{\frac{1}{N}} = \log_2 N.$$

Этот результат совпадает с формулой Хартли.

Однако формулу Хартли нельзя использовать, если вероятности событий разные. Пусть, например, детям раздают воздушные шарики разного цвета, причём 7 из каждых 10 шариков — зелёные. Какое количество информации содержится в сообщении

«Маша получила зелёный шарик»? Здесь формула Хартли неприменима, однако можно использовать формулу (1). Вероятность того, что Маше достался зелёный шарик, равна 7/10, а количество информации вычисляется как

$$I = \log_2 \frac{1}{\frac{7}{10}} = \log_2 \frac{10}{7} \text{ битов.}$$

Величина под знаком двоичного логарифма не является степенью двойки. Как её вычислить? Для этого используется свойство логарифма, позволяющее переходить к другому основанию, например, к десятичным или натуральным логарифмам, которые умеют вычислять калькуляторы:

$$I = \log_2 x = \frac{\lg x}{\lg 2} = \frac{\ln x}{\ln 2} \text{ битов.}$$

В данном случае получаем

$$I = \frac{\lg \frac{10}{7}}{\lg 2} = \frac{\ln \frac{10}{7}}{\ln 2} \approx 0{,}515 \text{ бита.}$$

## Вопросы и задания

1. Как вы понимаете термин «вероятность события»?
2. В каких случаях вероятность равна 1? Когда она равна 0?
3. Что неправильно в сообщении: «"Спартак" выиграет с вероятностью 200%»?

## Задачи

1. Вероятность появления символа @ в некотором тексте равна 0,125. Сколько битов информации несёт сообщение о том, что очередной символ текста — @?
2. В садке у рыбака сидят 2 окуня, 4 плотвы и 10 уклеек. Не смотря в садок, рыбак вытаскивает наугад одну рыбу. Какова вероятность того, что это будет плотва?
3. В пруду плавают 10 линей, 20 окуней и 70 карасей. Считаем, что они одинаково голодны и равномерно распределены по водоёму. Какова вероятность того, что первая рыба, пойманная рыбаком, будет линем? Окунем? Карасём?
4. Ученик задумал целое число от 1 до 100. Какова вероятность того, что это будет число в интервале от 21 до 30? От 31 до 55? Больше 25? Равно 25?

5. Ученик задумал целое число от –10 до 10. Какова вероятность того, что квадрат этого числа больше 25? Меньше 10? Равен 49?
6. В корзине лежат 8 чёрных шаров и 24 белых. Какова вероятность вытащить чёрный шар? Сколько битов информации несёт сообщение о том, что достали чёрный шар?
7. В корзине лежат 32 клубка шерсти, из них 4 красных. Сколько битов информации несёт сообщение о том, что достали клубок красной шерсти?
8. В коробке лежат 64 цветных карандаша. Сообщение о том, что достали белый карандаш, несёт 4 бита информации. Сколько белых карандашей было в коробке?
9. В ящике лежат чёрные и белые перчатки. Среди них 2 пары чёрных. Сообщение о том, что достали чёрные перчатки, несет 4 бита информации. Сколько всего пар перчаток было в ящике?
10. За контрольную работу в классе из 30 человек выставлено 6 пятёрок, 15 четвёрок, 8 троек и 1 двойка. Сколько битов информации несёт сообщение о том, что Иван Петров получил четвёрку?
11. В ящике лежат 20 шаров, из них 10 чёрных, 5 белых, 4 жёлтых и 1 красный. Сколько битов информации несёт сообщение о том, что достали белый шар?
12. За четверть ученик получил 20 оценок. Сообщение о том, что он вчера получил четвёрку, несёт 2 бита информации. Сколько четвёрок получил ученик за четверть?
13. В корзине лежат чёрные и белые шары. Среди них 18 чёрных шаров. Сообщение о том, что достали белый шар, несёт 2 бита информации. Сколько всего шаров было в корзине?
14. В алфавите языка племени тумба-юмба 4 буквы: гласные О и А, согласные Ш и Щ. Вероятности их появления в тексте: А — 0,35; О — 0,4; Ш — 0,1; Щ — 0,15. Сколько битов информации несёт сообщение о том, что очередной символ текста — согласная?

*15. Автобус № 25 ходит в 2 раза чаще, чем автобус № 13. Сообщение о том, что к остановке подошел автобус № 25, несет 4 бита информации. Сколько битов информации в сообщении «К остановке подошел автобус № 13»?

*16. В зоопарке 32 обезьяны живут в двух вольерах, А и Б. Одна из обезьян заболела. Сообщение «Заболела обезьяна из вольера А» содержит 4 бита информации. Сколько обезьян живут в вольере Б?

### Формула Шеннона

Информация играет для нас важную роль потому, что наше знание всегда неполно, в нём есть *неопределённость*. Эта неопределённость мешает нам решать свои задачи, принимать правиль-

ные решения. Полученная информация уменьшает («снимает») неопределённость, полностью или частично. Поэтому количество полученной информации можно оценить по величине уменьшения неопределённости:

$$\Delta H = H_{\text{нач}} - H_{\text{кон}},$$

где $H_{\text{нач}}$ — начальная неопределённость, а $H_{\text{кон}}$ — конечная (после получения сообщения). Если неопределённость полностью снимается, то $H_{\text{кон}} = 0$.

**Клод Шеннон (1916–2001)**

Чтобы оценить информацию с этой точки зрения, нужно как-то вычислить неопределённость, выразить её числом. Эту задачу решил в 1948 г. американский математик Клод Шеннон.

Пусть неопределённость состоит в том, что мы можем получить одно из $N$ возможных сообщений, причём известно, что вероятность получения сообщения с номером $i$ равна $p_i$.

Неопределённость знания об источнике данных вычисляется по формуле Шеннона

$$H = -\sum_{i=1}^{N} p_i \cdot \log_2 p_i = \sum_{i=1}^{N} p_i \cdot \log_2 \frac{1}{p_i}.$$

Величина $H$ часто называется *информационной энтропией*. С точки зрения математики это *среднее количество информации*, которую мы получаем при полном снятии неопределённости (когда выбран один из возможных вариантов).

Когда неопределённость наибольшая? Зададим вопрос «Идёт ли сейчас снег?» зимой и летом. Летом неопределённость очень маленькая, так как, скорее всего, снега нет, ситуация ясна. Зимой же неопределённость велика, потому что снег может идти или не идти примерно с равной вероятностью.

Перейдём к числам. Будем считать, что вероятность снега зимой равна $p_1 = 0{,}5$. Чему равна вероятность $p_2$ того, что снега нет? «Здравый смысл» подсказывает, что $p_2 = 0{,}5$ (остальные 50%). Математики говорят, что два события, «Снег идёт» и «Снега нет», составляют **полную систему**. Это значит, что обязательно случится какое-нибудь одно из этих событий, и при этом другое точно не произойдёт. Слово «обязательно» означает, что вероятность этих двух событий в сумме равна 1.

Сумма вероятностей всех событий, составляющих полную систему, равна 1.

Для «зимнего» случая количество информации при получении сообщений «Снег идёт» и «Снега нет» одинаковое, потому что их вероятности одинаковые:

$$I_1 = I_2 = \log_2 \frac{1}{0{,}5} = 1 \text{ бит}.$$

Неопределённость, вычисленная по формуле Шеннона, также равна 1 биту:

$$H = p_1 \cdot \log_2 \frac{1}{p_1} + p_2 \cdot \log_2 \frac{1}{p_2} = 0{,}5 \cdot \log_2 \frac{1}{0{,}5} + 0{,}5 \cdot \log_2 \frac{1}{0{,}5} = \log_2 2 = 1 \text{ бит}.$$

Для лета будем считать вероятность выпадения снега равной $p_1 = 0{,}0001$. Тогда вероятность того, что снега нет, равна $p_2 = 1 - 0{,}0001 = 0{,}9999$. Сообщения «Снег идёт» и «Снега нет» несут разное количество информации:

$$I_1 = \log_2 \frac{1}{0{,}0001} = 13{,}29 \text{ бита}, \quad I_2 = \log_2 \frac{1}{0{,}9999} = 0{,}00014 \text{ бита},$$

а неопределённость (среднее количество информации) равна

$$H = 0{,}0001 \cdot \log_2 \frac{1}{0{,}0001} + 0{,}9999 \cdot \log_2 \frac{1}{0{,}9999} \approx 0{,}0015 \text{ бита}.$$

Мы получили то, что ожидали: зимой неопределённость в ответе на вопрос «Идёт ли сейчас снег?» значительно больше, чем летом. Можно предположить (и это действительно так), что неопределённость наибольшая в том случае, когда вероятности всех событий равны.

Летом эта неопределённость очень близка к нулю, поэтому можно предположить (и это также верно), что она стремится к нулю, если вероятность одного из двух событий стремится к нулю.

На рисунке 1.1 построена зависимость величины неопределённости $H$ от вероятности первого события $p_1$. При этом вероятность второго события определяется как $p_2 = 1 - p_1$, так как они составляют полную систему. Главный вывод этого примера таков:

Неопределённость наибольшая для случая, когда все события равновероятны.

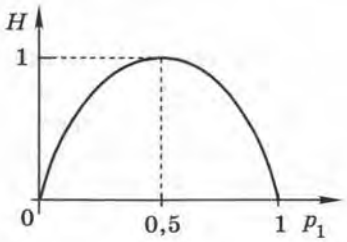



**Рис. 1.1**

При этом вероятность каждого из $N$ событий равна $p = \frac{1}{N}$, поэтому по формуле Шеннона

$$H = \sum_{i=1}^{N} p \cdot \log_2 \frac{1}{p} = \sum_{i=1}^{N} \frac{1}{N} \cdot \log_2 N = \log_2 N.$$

Отсюда следует, что:

При равновероятных событиях неопределённость совпадает с количеством информации, вычисленной по формуле Хартли.

## Вопросы и задания

1. В чём заключается неопределённость?
2. Как связана неопределённость с информацией, которую мы получаем при сообщении об отдельных событиях?
3. Что такое полная система событий?
4. Что можно сказать о вероятностях событий, входящих в полную систему?
5. В каком случае неопределённость наибольшая?
6. В каком случае неопределённость стремится к нулю?
7. В каком случае неопределённость совпадает с количеством информации, вычисленным по формуле Хартли?

### Задачи

1. Из аэропорта Куково можно улететь на самолетах Ту-154, АН-148, Боинг-737. Вероятность полёта на самолёте Ту-154 равна 0,6; вероятность полёта на Боинге-737 равна 0,1. Чему равна вероятность полёта на АН-148?
2. В коробке 3 красных карандаша и 7 синих. Чему равна неопределённость при выборе наугад одного карандаша из коробки?
3. На улице Строителей из 20 домов 6 деревянных, 8 сделаны из кирпича, а оставшиеся — из железобетонных плит. Чему равна неопределённость ответа на вопрос «Из чего сделан дом № 16 на улице Строителей»?

## § 2
## Передача данных

### Скорость передачи данных

**Скорость передачи данных** — это количество битов (байтов, Кбайт и т. д.), которое передаётся по каналу связи за единицу времени (например, за 1 с).

Пропускная способность любого реального канала связи ограничена. Это значит, что есть некоторая наибольшая возможная скорость передачи данных, которую принципиально невозможно превысить.

Основная единица измерения скорости — биты в секунду (бит/с, англ. **bps** — *bits per second*). Для характеристики быстродействующих каналов применяют единицы измерения с десятичными приставками (а не двоичными, как при измерении количества информации): килобиты в секунду (1 кбит/с = 1000 бит/с), мегабиты в секунду (1 Мбит/с = $10^6$ бит/с) и гигабиты в секунду (1 Гбит/с = $10^9$ бит/с); иногда используют байты в секунду (байт/с) и мегабайты в секунду (1 Мбайт/с = $10^6$ байт/с).

Информационный объём $I$ данных, переданных по каналу за время $t$, вычисляется по формуле $I = v \cdot t$, где $v$ — скорость передачи данных. Например, если скорость передачи данных равна 512 000 бит/с, за 1 минуту можно передать файл объёмом

512 000 бит/с · 60 с = 30 720 000 битов = 3 840 000 байтов = 3750 Кбайт.

**Обнаружение ошибок**

В реальных каналах связи всегда присутствуют помехи, искажающие сигнал. В некоторых случаях ошибки допустимы, например, при прослушивании радиопередачи через Интернет небольшое искажение звука не мешает понимать речь. Однако чаще всего требуется обеспечить точную передачу данных. Для этого в первую очередь нужно определить факт возникновения ошибки и, если это произошло, передать блок данных ещё раз.

Представьте себе, что получена цепочка нулей и единиц 1010101110, причём все биты независимы. В этом случае нет абсолютно никакой возможности определить, верно ли передана последовательность. Поэтому необходимо вводить **избыточность** в передаваемое сообщение (включать в него «лишние» биты) только для того, чтобы обнаружить ошибку.

Простейший вариант — добавить 1 бит в конце блока данных, который будет равен 1, если в основном сообщении нечётное число единиц, и равен 0 для сообщения с чётным числом единиц. Этот дополнительный бит называется **битом чётности**. Бит чётности используется при передаче данных в сетях, проверка чётности часто реализуется аппаратно (с помощью электроники).

Например, пусть требуется передать два бита данных. Возможны всего 4 разных сообщения: 00, 01, 10 и 11. Первое и четвёртое из них содержат чётное число единиц (0 и 2), значит, бит чётности для них равен 0. Во втором и третьем сообщениях нечётное число единиц (1), поэтому бит чётности будет равен 1. Таким образом, сообщения с добавленным битом чётности будут выглядеть так:

$$00\underline{0},\ 01\underline{1},\ 10\underline{1},\ 11\underline{0}.$$

Первые два бита несут полезную информацию, а третий (подчёркнутый) — вспомогательный, он служит только для обнаружения ошибки. Обратим внимание, что каждое из этих трёхбитных сообщений содержит чётное число единиц.

Подумаем, сколько ошибок может обнаружить такой метод. Если при передаче неверно передан только один из битов, количество единиц в сообщении стало нечётным, это и служит признаком ошибки при передаче. Однако исправить ошибку нельзя, потому что непонятно, в каком именно разряде она случилась.

Если же изменилось два бита, чётность не меняется, и такая ошибка не обнаруживается. В длинной цепочке применение бита

чётности позволяет обнаруживать нечётное число ошибок (1, 3, 5, ...), а ошибки в чётном количестве разрядов остаются незамеченными.

Контроль с помощью бита чётности применяется для небольших блоков данных (чаще всего — для каждого отдельного байта) и хорошо работает тогда, когда отдельные ошибки при передаче независимы одна от другой и встречаются редко.

Для обнаружения искажений в передаче файлов, когда может сразу возникнуть множество ошибок, используют другой метод — вычисляют контрольную сумму с помощью какой-нибудь хэш-функции (вспомните материал учебника для 10 класса). Чаще всего для этой цели применяют алгоритмы **CRC** (англ. *Cyclic Redundancy Code* — циклический избыточный код), а также криптографические хэш-функции MD5, SHA-1 и другие. Если контрольная сумма блока данных, вычисленная приёмником, не совпадает с контрольной суммой, записанной передающей стороной, то произошла ошибка.

**Помехоустойчивые коды**

Значительно сложнее исправить ошибку сразу (без повторной передачи), однако в некоторых случаях и эту задачу удаётся решить. Для этого требуется настолько увеличить избыточность кода (добавить «лишние» биты), что небольшое число ошибок всё равно позволяет достаточно уверенно распознать переданное сообщение. Например, несмотря на помехи в телефонной линии, обычно мы легко понимаем собеседника. Это значит, что речь обладает достаточно большой избыточностью, и это позволяет исправлять ошибки «на ходу».

Пусть, например, нужно передать один бит, 0 или 1. Утроим его, добавив ещё два бита, совпадающих с первым. Таким образом, получаются два «правильных» сообщения:

000 и 111.

Теперь посмотрим, что получится, если при передаче одного из битов сообщения 000 произодёт ошибка и приёмник получит искажённое сообщение 001. Заметим, что оно отличается одним битом от 000 и двумя битами от второго возможного варианта — 111. Значит, скорее всего, произошла ошибка в последнем бите и сообщение нужно исправить на 000. Если приёмник получил 101, можно точно сказать, что произошла ошибка, однако попытка исправить её приведёт к неверному варианту, так как ближайшая

«правильная» последовательность — это 111. Таким образом, такой код *обнаруживает* одну или две ошибки. Кроме того, он позволяет *исправить* (!) одну ошибку, т. е. является помехоустойчивым.

**Помехоустойчивый код** — это код, который позволяет исправлять ошибки, если их количество не превышает некоторого уровня.

Выше мы фактически применили понятие «расстояния» между двумя кодами. В теории передачи информации эта величина называется расстоянием Хэмминга в честь американского математика Р. Хэмминга.

**Расстояние Хэмминга** — это количество позиций, в которых различаются два закодированных сообщения одинаковой длины.

Например, расстояние $d$ между кодами 001 и 100 равно

$$d(\underline{0}0\underline{1}, \underline{1}0\underline{0}) = 2,$$

потому что они различаются в двух битах (эти биты подчёркнуты). В приведённом выше примере расстояние между «правильными» последовательностями (*словами*) равно $d(000, 111) = 3$. Такой код позволяет обнаружить одну или две ошибки и исправить одну ошибку.

В общем случае, если минимальное расстояние между «правильными» словами равно $d$, можно обнаружить от 1 до $d - 1$ ошибок, потому что при этом полученный код будет отличаться от всех допустимых вариантов. Для исправления $r$ ошибок необходимо, чтобы выполнялось условие

$$d \geq 2r + 1.$$

Это значит, что слово, в котором сделано $r$ ошибок, должно быть ближе к исходному слову (из которого оно получено искажением), чем к любому другому.

Рассмотрим более сложный пример. Пусть нужно передавать три произвольных бита, обеспечив обнаружение двух любых ошибок и исправление одной ошибки. В этом случае можно использо-

вать, например, такой код с тремя контрольными битами (они подчёркнуты):

| | |
|---|---|
| 000 000 | 100 101 |
| 001 111 | 101 010 |
| 010 011 | 110 110 |
| 011 100 | 111 001 |

Расстояние Хэмминга между любыми двумя словами в таблице не менее 3, поэтому код обнаруживает две ошибки и позволяет исправить одну. Как же вычислить ошибочный бит?

Предположим, что было получено кодовое слово 011011. Определив расстояние Хэмминга до каждого из «правильных» слов, находим единственное слово 010011, расстояние до которого равно 1 (расстояния до остальных слов больше). Значит, скорее всего, это слово и было передано, но исказилось из-за помех.

На практике используют несколько более сложные коды, которые называются **кодами Хэмминга**. В них информационные и контрольные биты перемешаны, и за счёт этого можно сразу, без перебора, определить номер бита, в котором произошла ошибка. Наиболее известен семибитный код, в котором 4 бита — это данные, а 3 бита — контрольные. В нём минимальное расстояние между словами равно 3, поэтому он позволяет обнаружить две ошибки и исправить одну.

## Вопросы и задания

1. В каких единицах измеряют скорость передачи данных?
2. Почему для любого канала связи скорость передачи данных ограничена?
3. Как вычисляется информационный объём данных, который можно передать за некоторое время?
4. В каких случаях при передаче данных допустимы незначительные ошибки?
5. Что такое избыточность сообщения? Для чего её можно использовать? Приведите примеры.
6. Как помехи влияют на передачу данных?
7. Что такое бит чётности? В каких случаях с помощью бита чётности можно обнаружить ошибку, а в каких — нельзя?
8. Можно ли исправить ошибку, обнаружив неверное значение бита чётности?
9. Для чего используется метод вычисления контрольной суммы?

10. Какой код называют помехоустойчивым?
11. Каково должно быть расстояние Хэмминга между двумя любыми кодами, чтобы можно было исправить 2 ошибки?
12. Как исправляется ошибка при использовании помехоустойчивого кода?
13. Сколько ошибок обнаруживает 7-битный код Хэмминга, описанный в конце параграфа, и сколько ошибок он позволяет исправить?

**Подготовьте сообщение**

а) «Алгоритмы CRC»

б) «Коды Хемминга»

## Задачи

1. Через соединение со скоростью 128 000 бит/с передают файл размером 625 Кбайт. Определите время передачи файла в секундах.
2. Передача файла через соединение со скоростью 512 000 бит/с заняла 1 минуту. Определите размер файла в килобайтах.
3. Скорость передачи данных равна 64 000 бит/с. Сколько времени займёт передача файла объёмом 375 Кбайт по этому каналу?
4. У Васи есть доступ к Интернету по высокоскоростному одностороннему радиоканалу, обеспечивающему скорость получения им информации 256 000 бит/с. У Пети нет скоростного доступа к Интернету, но есть возможность получать информацию от Васи по низкоскоростному телефонному каналу со средней скоростью 32 768 бит/с. Петя договорился с Васей, что тот будет скачивать для него данные объёмом 5 Мбайт по высокоскоростному каналу и ретранслировать их Пете по низкоскоростному каналу. Компьютер Васи может начать ретрансляцию данных не раньше, чем им будут получены первые 375 Кбайт этих данных. Каков минимально возможный промежуток времени (в секундах) с момента начала скачивания Васей данных до полного их получения Петей?
5. Определите максимальный размер файла в килобайтах, который может быть передан за 8 минут со скоростью 32 000 бит/с.
6. Сколько секунд потребуется, чтобы передать 400 страниц текста, состоящего из 30 строк по 60 символов каждая по линии со скоростью 128 000 бит/с, при условии что каждый символ кодируется 1 байтом?
7. Передача текстового файла по каналу связи со скоростью 128 000 бит/с заняла 1,5 мин. Определите, сколько страниц содержал переданный текст, если известно, что он был представлен в 16-битной кодировке UNICODE, а на одной странице — 400 символов.

8. Передача текстового файла через соединение со скоростью 64 000 бит/с заняла 10 с. Определите, сколько страниц содержал переданный текст, если известно, что он был представлен в 16-битной кодировке UNICODE и на каждой странице — 400 символов.
9. Сколько секунд потребуется, чтобы передать цветное растровое изображение размером 1280 × 800 пикселей по линии со скоростью 256 000 бит/с при условии, что цвет каждого пикселя кодируется 24 битами?
10. Сколько секунд потребуется, чтобы передать цветное растровое изображение размером 1000 × 800 пикселей по линии со скоростью 128 000 бит/с при условии, что цвет каждого пикселя кодируется 24 битами?
11. Через некоторый канал связи за 2 минуты был передан файл, размер которого — 3750 Кбайт. Определите минимальную скорость, при которой такая передача возможна.
12*. Модем, передающий информацию со скоростью 256 000 бит/с, передал файл с несжатой стереофонической музыкой за 2 минуты 45 секунд. Найдите разрядность кодирования этой музыки, если известно, что её продолжительность составила 1 минуту и оцифровка производилась с частотой 22 000 Гц.
13. Найдите расстояние между кодами 11101 и 10110, YUIX и YAIY.
14. Найдите все пятизначные двоичные коды, расстояние от которых до кода 11101 равно 1. Сколько всего может быть таких слов для $n$-битного кода?
15. Для передачи восьмеричных чисел используется помехоустойчивый шестибитный код, приведённый в тексте параграфа. Раскодируйте сообщение, исправив ошибки:

    001 101 011 011 011 101 110 101 101 011

## § 3
## Сжатие данных

### Основные понятия

Для того чтобы сэкономить место на внешних носителях (жёстких дисках, флэш-дисках) и ускорить передачу данных по компьютерным сетям, нужно сжать данные — уменьшить информационный объём, сократить длину двоичного кода. Это можно сделать, устранив **избыточность** использованного кода.

Например, пусть текстовый файл объёмом 10 Кбайт содержит всего четырех различных символа: латинские буквы «A», «B», «C» и пробел. Вы знаете, что для кодирования одного из четырёх возможных вариантов достаточно 2 битов, поэтому использовать для его передачи обычное 8-битное кодирование символов невыгодно. Можно присвоить каждому из четырёх символов двухбитные коды, например, так:

A — 00, B — 01, C — 10, пробел — 11.

Тогда последовательность «ABA CABABA», занимающая 10 байтов в однобайтной кодировке, может быть представлена как цепочка из 20 битов:

00010011100001000100

Таким образом, нам удалось уменьшить информационный объём текста в 4 раза, и передаваться он будет в 4 раза быстрее. Однако непонятно, как раскодировать это сообщение, ведь получатель не знает, какой код был использован. Выход состоит в том, чтобы включить в сообщение служебную информацию — заголовок, в котором каждому коду будет сопоставлен ASCII-код символа. Условимся, что первый байт заголовка — это количество используемых символов $N$, а следующие $N$ байтов — это ASCII-коды этих символов. В данном случае заголовок занимает 5 байтов и выглядит так:

| $00000100_2$ | $01000001_2$ | $01000010_2$ | $01000011_2$ | $00100000_2$ |
|---|---|---|---|---|
| 4 символа | A (код 65) | B (код 66) | C (код 67) | пробел (код 32) |

Файл, занимающий 10 Кбайт в 8-битной кодировке, содержит 10 240 символов. В сжатом виде каждый символ кодируется двумя битами, кроме того, есть 5-байтный заголовок. Поэтому сжатый файл будет иметь объём

$5 + 10240 \cdot 2/8$ байтов = 2565 байтов.

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

В данном случае удалось сжать файл почти в 4 раза, коэффициент сжатия равен

$$k = 10\ 240/2565 \approx 4.$$

Если принимающая сторона «знает» формат файла (заголовок + закодированные данные), она сможет восстановить в точности его исходный вид. Такое сжатие называют *сжатием без потерь*. Оно используется для упаковки текстов, программ, данных, которые ни в коем случае нельзя искажать.

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

За счёт чего удалось сжать файл? Только за счёт того, что в файле была некоторая закономерность, избыточность — использовались только 4 символа вместо полного набора.

### Алгоритм RLE

При сжатии данных, в которых есть цепочки одинаковых кодов, можно применять ещё один простой алгоритм, который называется **кодированием цепочек одинаковых** символов (англ. **RLE** — *Run Length Encoding*). Представим себе файл, в котором записаны сначала 100 русских букв «А», а потом — 100 букв «Б»:

| 1 | 100 | 101 | 200 |
|---|---|---|---|
| АА……АА | | ББ……ББ | |

При использовании алгоритма RLE сначала записывается количество повторений первого символа, затем — сам первый символ, затем — количество повторений второго символа, затем — второй символ и т. д. В данном случае весь закодированный файл занимает 4 байта:

| $01100100_2$ | $11000000_2$ | $01100100_2$ | $11000001_2$ |
|---|---|---|---|
| 100 | А (код 192) | 100 | Б (код 193) |

Таким образом, мы сжали файл в 50 раз за счёт того, что в нём снова была *избыточность* — цепочки одинаковых символов. Это

*сжатие без потерь*, потому что, зная алгоритм упаковки, исходные данные можно точно восстановить из кода.

Очевидно, что такой подход будет приводить к *увеличению* (в 2 раза) объема данных в том случае, когда в файле нет соседних одинаковых символов. Чтобы улучшить результаты RLE-кодирования даже в этом наихудшем случае, алгоритм модифицировали следующим образом. Упакованная последовательность содержит управляющие байты, за каждым управляющим байтом следует один или несколько байтов данных. Если старший бит управляющего байта равен 1, то следующий за управляющим байт данных при распаковке нужно повторить столько раз, сколько записано в оставшихся 7 битах управляющего байта. Если же старший бит управляющего байта равен 0, то надо взять несколько следующих байтов данных без изменения. Сколько именно — записано в оставшихся 7 битах управляющего байта. Например, управляющий байт $10000111_2$ говорит о том, что следующий за ним байт надо повторить 7 раз, а управляющий байт $00000100_2$ — о том, что следующие за ним 4 байта надо взять без изменений. Например, последовательность

| $10001111_2$ | $11000000_2$ | $00000010_2$ | $11000001_2$ | $11000010_2$ |
|---|---|---|---|---|
| повтор 15 | А (код 192) | 2 | Б (код 193) | В (код 194) |

распаковывается в 17 символов: АААААААААААААААББ.

Алгоритм RLE успешно использовался для сжатия рисунков, в которых большие области закрашены одним цветом, и некоторых звуковых данных. Сейчас вместо него применяют более совершенные, но более сложные методы. Один из них (алгоритм Хаффмана) мы рассмотрим далее. Алгоритм RLE используется, например, на одном из этапов кодирования рисунков в формате JPEG. Возможность использования RLE-сжатия есть также в формате BMP (для рисунков с палитрой 16 или 256 цветов).

### Префиксные коды

Вспомните азбуку Морзе, в которой для уменьшения длины сообщения используется неравномерный код — часто встречающиеся буквы (А, Е, М, Н, Т) кодируются короткими последовательностями, а редко встречающиеся — более длинными. Такой код можно представить в виде структуры, которая называется **деревом** (вспомните дерево каталогов).

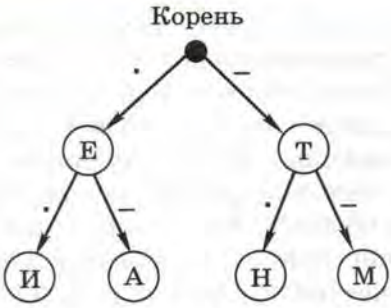



**Рис. 1.2**

На рисунке 1.2 показано неполное дерево кода Морзе, построенное только для символов, коды которых состоят из одного и двух знаков (точек и тире). Дерево состоит из узлов (большая чёрная точка и кружки с символами алфавита) и соединяющих их направленных рёбер, стрелки указывают направление движения. Верхний узел (в который не входит ни одна стрелка) называется **корнем дерева**. Из корня и из всех промежуточных узлов (кроме конечных узлов — **листьев**) выходят две стрелки, левая помечена точкой, а правая — знаком «тире». Чтобы найти код символа, нужно пройти по стрелкам от корня дерева к нужному узлу, выписывая метки стрелок, по которым мы переходим. В дереве нет циклов (замкнутых путей), поэтому кодовое слово для каждого символа определяется единственным образом. По этому дереву можно построить такие кодовые слова:

Е • И •• А •– Т – Н –• М ––

Это неравномерный код, в нём символы имеют коды разной длины. При этом всегда возникает проблема разделения последовательности на отдельные кодовые слова. В коде Морзе она решена с помощью символа-разделителя — паузы. Однако можно не вводить дополнительный символ, если выполняется **условие Фано**: ни одно из кодовых слов не является началом другого кодового слова. Это позволяет однозначно раскодировать сообщение в реальном времени, по мере получения очередных символов.

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

Для использования этой идеи в компьютерной обработке данных нужно было разработать алгоритм построения префиксного

кода. Впервые эту задачу решили, независимо друг от друга, американские математики и инженеры Клод Шеннон и Роберт Фано **(код Шеннона–Фано)**. Они использовали *избыточность* сообщений, состоящую в том, что символы в тексте имеют разные частоты встречаемости. В этом случае для построения кода нужно читать данные исходного файла два раза: на первом проходе определяется частота встречаемости каждого символа, затем строится код с учётом этих данных, и на втором проходе символы текста заменяются на их коды.

Пусть, например, текст состоит только из букв «О», «Е», «Н», «Т» и пробела. Известно, сколько раз они встретились в тексте: пробел — 179, О — 89, Е — 72, Н — 53 и Т — 50 раз. Делим символы на 2 группы так, чтобы общее количество найденных в тексте символов первой группы было примерно равно общему количеству символов второй группы. В нашем случае лучший вариант — это объединить пробел и букву Т в первую группу (сумма 179 + 50 = 229), а остальные символы — во вторую (сумма 89 + 72 + 53 = 214).

Символы первой группы будут иметь коды, начинающиеся с 0, а остальные — с 1. В первой группе всего два символа, у одного из них, например у пробела, вторая цифра кода будет 0 (и полный код 00), а у второго — 1 (код буквы Т — 01) .

Во второй группе три символа, поэтому продолжаем деление на две группы, примерно равные по количеству символов в тексте. В первую выделяем одну букву, которая чаще всего встречается — это буква О (её код будет 10), а во вторую — буквы Е и Н (они получают коды 110 и 111). Код Шеннона–Фано, построенный для этого случая, можно нарисовать в виде дерева (рис. 1.3).

Символ ␣ на этой схеме обозначает пробел.

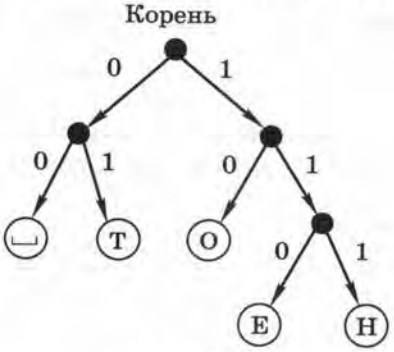



**Рис. 1.3**

Легко проверить, что для этого кода выполняется условие Фано. Это можно сразу определить по построенному дереву. В нём все символы располагаются в листьях, а не в промежуточных узлах. Это значит, что «по пути» от корня дерева до любого символа никаких других символов в промежуточных узлах не встречается (сравните с деревом кода Морзе).

Для раскодирования очередного символа последовательности мы просто спускаемся от корня дерева, выбирая левую ветку, если очередной бит — 0, и правую, если этот бит равен 1. Дойдя до листа дерева, мы определяем символ, а затем снова начинаем с корня дерева, чтобы раскодировать следующий символ, и т. д. Например, пусть получена последовательность

01100110001101111001.

Результат ее раскодирования — «ТОТО ЕНОТ».

**Алгоритм Хаффмана**

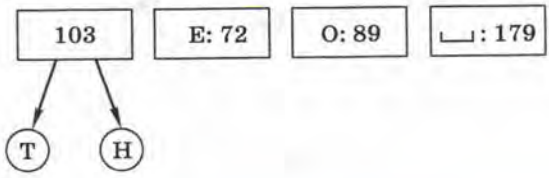
**Дэвид Хаффман (1925–1999)**

Было доказано, что в некоторых случаях кодирование Шеннона–Фано дает неоптимальное решение, и можно построить код, который ещё больше уменьшит длину кодовой последовательности. Например, в предыдущем примере (код Шеннона–Фано) пробел и буква «Т» имеют коды одинаковой длины, хотя буква «Т» встречается в 3,5 раза реже пробела. Через несколько лет Дэвид Хаффман, ученик Фано, разработал новый алгоритм кодирования и доказал его оптимальность.

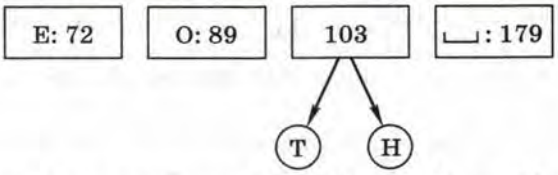
Построим код Хаффмана для того же самого примера. Сначала отсортируем буквы по увеличению частоты встречаемости:

| Т: 50 | Н: 53 | Е: 72 | О: 89 | ␣: 179 |
|---|---|---|---|---|

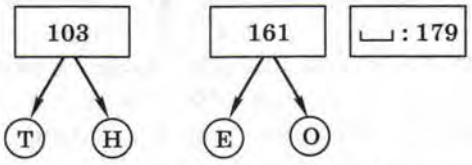
Затем берём две самые первые буквы, они становятся листьями дерева, а в узел, с которым они связаны, записываем сумму их частот:



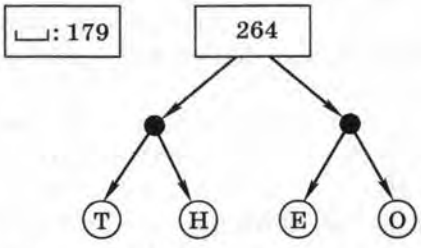
Таким образом, буквы, которые встречаются реже всего, получили самый длинный код. Снова сортируем буквы по возрастанию частоты, но для букв «Т» и «Н» используется их суммарная частота:




Повторяем ту же процедуру для букв «Е» и «О», частоты которых оказались минимальными, и сортируем по возрастанию частоты:




Теперь объединяем уже не отдельные буквы, а пары, и снова сортируем:




На последнем шаге остаётся объединить символ «пробел» с деревом, которое построено для остальных символов. У каждой стрелки, идущей влево от какого-то узла, ставим код 0, а у каждой стрелки, идущей вправо — код 1 (рис. 1.4).

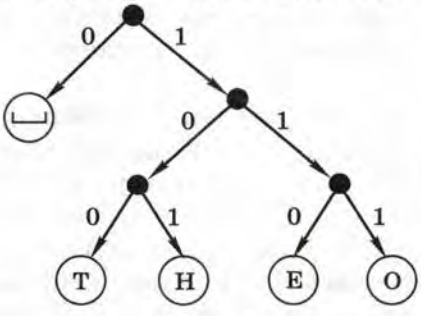



**Рис. 1.4**

По этому дереву, спускаясь от корня к листьям-символам, получаем коды Хаффмана, обеспечивающие оптимальное сжатие с учётом частоты встречаемости символов:

␣ — 0, Т — 100, О — 111, Е — 110, Н — 101.

Легко проверить, что этот код также удовлетворяет условию Фано.

Сравним эффективность рассмотренных выше методов. Для алфавита из 5 символов при равномерном кодировании нужно использовать 3 бита на каждый символ, так что общее число битов в сообщении равно $(179 + 89 + 72 + 53 + 50) \cdot 3 = 1329$ битов. При кодировании методом Шеннона–Фано получаем $179 \cdot 2 + 89 \cdot 2 + 72 \cdot 3 + 53 \cdot 3 + 50 \cdot 2 = 1011$ битов, коэффициент сжатия составляет $1329/1011 \approx 1{,}31$. Использование кода Хаффмана даёт последовательность длиной $179 \cdot 1 + 89 \cdot 3 + 72 \cdot 3 + 53 \cdot 3 + 50 \cdot 3 = 971$ бит, коэффициент сжатия равен 1,37. В сравнении со случаем, когда для передачи используется однобайтный код (8 битов на символ), выигрыш получается ещё более весомым: кодирование Хаффмана сжимает данные примерно в 3,65 раза. Обратим внимание, что сжать данные удалось за счёт *избыточности*: мы использовали тот факт, что некоторые символы встречаются чаще, а некоторые — реже.

Алгоритм кодирования Хаффмана и сейчас широко применяется благодаря своей простоте, высокой скорости кодирования и отсутствию патентных ограничений (он может быть использован свободно). Например, он применяется на некоторых этапах при сжатии рисунков (в формате JPEG) и звуковой информации (в формате MP3).

### Сжатие с потерями

Выше мы рассмотрели методы *сжатия без потерь*, при которых можно в точности восстановить исходную информацию, зная алгоритм упаковки.

В некоторых случаях небольшие неточности в передаче данных несущественно влияют на решение задачи. Например, небольшие помехи в телефонном разговоре или в радиопередаче, которая транслируется через Интернет, не мешают пониманию речи. Некоторое дополнительное размытие уже и без того размытого изображения (фотографии) непрофессионал даже не заметит. Ещё в большей степени это относится к видеофильмам. Поэтому

иногда можно немного пожертвовать качеством изображения или звука ради того, чтобы значительно сократить объём данных. В этих случаях используется *сжатие с потерями*.

**Сжатие с потерями** — это такое уменьшение объёма закодированных данных, при котором распакованный файл может отличаться от оригинала.

Самые известные примеры сжатия с потерями — это алгоритмы JPEG (для изображений), MP3 (для упаковки звука) и все алгоритмы упаковки видеофильмов (MJPEG, MPEG4, DivX, XviD).

Простейший метод сжатия рисунков — снижение глубины цвета. На рисунке 1.5 показаны три изображения с различной глубиной цвета. Третье из них занимает при кодировании в 4 раза меньше места, чем первое, хотя воспринимается уже с трудом.

8 битов на пиксель (256 цветов) | 4 бита на пиксель (16 цветов) | 2 бита на пиксель (4 цвета)

**Рис. 1.5**

**Алгоритм JPEG** (англ. *Joint Photographic Experts Group* — объединенная группа экспертов по фотографии) — один из наиболее эффективных методов сжатия с потерями для изображений. Он довольно сложен, поэтому мы опишем только основные идеи этого алгоритма.

Из классической RGB-модели (красный, зелёный, синий) цвет переводится в **модель YCbCr**, где Y — яркость, Cb — «синева», Cr — «краснота»:

$$Y = 0{,}299 \cdot R + 0{,}587 \cdot G + 0{,}114 \cdot B,$$
$$Cb = 128 - 0{,}1687 \cdot R - 0{,}3313 \cdot G + 0{,}5 \cdot B,$$
$$Cr = 128 + 0{,}5 \cdot R - 0{,}4187 \cdot G - 0{,}0813 \cdot B,$$

В этих формулах R, G и B обозначают красную, зелёную и синюю составляющие цвета в RGB-модели. Чёрно-белое изображение (точнее, оттенки серого), содержит только Y-составляющую, компоненты цвета Cb и Cr равны 128 и не несут полезной информации.

Основная идея сжатия основана на том, что человеческий глаз более чувствителен к изменению яркости, т. е. составляющей Y. Поэтому на синюю и красную составляющие, Cb и Cr, приходится меньше полезной информации, и на этом можно сэкономить. Рассмотрим квадрат размером 2 × 2 пикселя. Можно, например, сделать так: запомнить Y-компоненту для всех пикселей изображения (4 числа) и по одной компоненте Cb и Cr на все 4 пикселя (рис. 1.6).

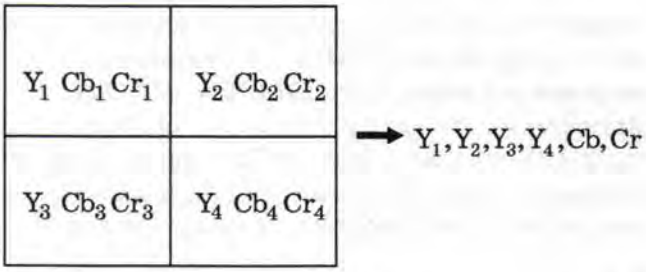



**Рис. 1.6**

Для определения сохраняемых значений Cb и Cr можно использовать, например, усреднение — принять

$$Cb = (Cb_1 + Cb_2 + Cb_3 + Cb_4)/4,\ \ Cr = (Cr_1 + Cr_2 + Cr_3 + Cr_4)/4.$$

Таким образом, вместо 12 чисел мы должны хранить всего 6, данные сжаты в 2 раза. Однако мы не сможем восстановить обратно исходный рисунок, потому что информация и том, какие были значения Cb и Cr для каждого пикселя, безвозвратно утеряна, есть только некоторые средние значения. При выводе такого изображения на экран можно, например, считать, что все 4 пикселя имели одинаковые значения Cb и Cr. Более сложные алгоритмы используют информацию о соседних блоках, сглаживая резкие переходы при изменении этих составляющих. Поэтому при кодировании с помощью алгоритма JPEG изображение несколько размывается.

Но это еще не всё. После этого к оставшимся коэффициентам применяется сложное *дискретное косинусное преобразование*, с помощью которого удаётся выделить информацию, которая мало влияет на восприятие рисунка человеком, и отбросить её. На последнем этапе для сжатия оставшихся данных используются алгоритмы RLE и Хаффмана.

На рисунке 1.7 вы видите результат сжатия изображения шарика в формате JPEG с разным качеством.

Качество 100 (8400 байтов) | Качество 50 (3165 байтов) | Качество 0 (1757 байтов) | Фрагмент (качество 0)

**Рис. 1.7**

Рисунки, сохранённые с качеством 100 и 50, практически не отличаются внешне, однако второй из них занимает в 2,65 раза меньше места. При снижении качества до минимального явно заметны искажения (*артефакты*), вызванные потерей информации при сжатии. Кажется, что рисунок построен из квадратов размером 8 × 8 пикселей (в действительности именно такие квадраты использует алгоритм в качестве базовых ячеек). Кроме того, искажения особенно заметны там, где происходит резкий переход от одного цвета к другому.

Тот же самый рисунок был сохранён в формате **BMP** без сжатия и с **RLE-сжатием**, а также в форматах **GIF** (сжатие без потерь с помощью **алгоритма LZW**) и **PNG** (сжатие без потерь с помощью **алгоритма DEFLATE**). Размеры полученных файлов (в килобайтах) сравниваются на диаграмме (рис. 1.8).

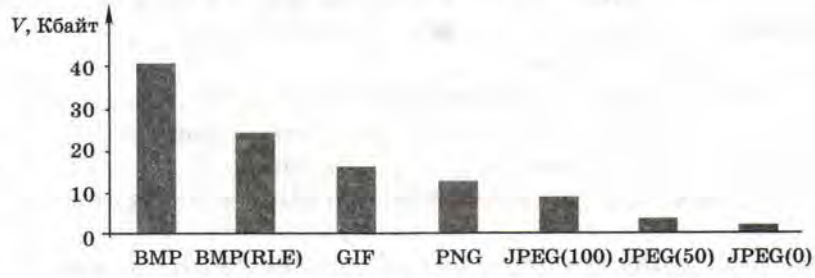



**Рис. 1.8**

По диаграмме видно, что наибольшее сжатие (наименьший объём файла) обеспечивается алгоритмом JPEG. Заметим, что особенность этого рисунка — плавные переходы между цветами.

Теперь рассмотрим другой рисунок, в котором у объектов есть большие области одного цвета и чёткие границы у объектов. Здесь ситуация несколько меняется (рис. 1.9).

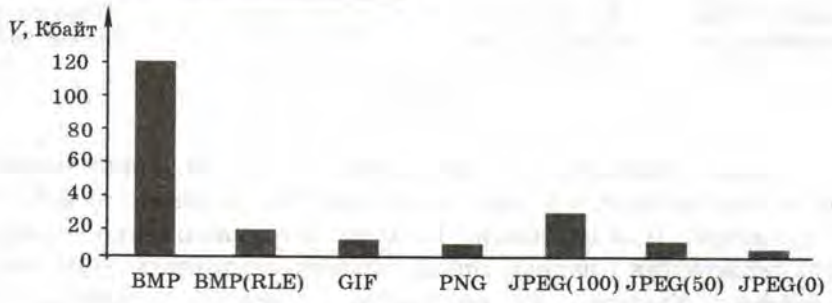



**Рис. 1.9**

Форматы GIF и PNG обеспечивают степень сжатия, сравнимую с алгоритмом JPEG среднего качества. Учитывая, что JPEG приводит к искажению изображения, а GIF и PNG — нет, для таких рисунков не рекомендуется использовать формат JPEG.

Другой известный пример сжатия с потерями — **алгоритм MPEG-1 Layer 3**, который более известен как **MP3**. В нём отбрасываются некоторые компоненты звука, которые практически неразличимы для большинства людей. Такой подход называется *кодированием восприятия*, потому что он основан на особенностях восприятия звука человеком.

Одна из главных характеристик сжатия звукового файла — **битрейт** (англ. *bitrate* — темп поступления битов). Битрейт — это число битов, используемых для кодирования 1 с звука. Формат MP3 поддерживает разные степени сжатия, с битрейтом от 8 до 320 Кбит/с.

Звук, закодированный с битрейтом 256 Кбит/с и выше, даже профессионал вряд ли отличит от звучания компакт-диска. Вспоминая материал учебника 10 класса, можно подсчитать, что 1 секунда звука занимает на компакт-диске (частота дискретизации 44 кГц, глубина кодирования 16 битов, стерео)

$2 \cdot 88\,000 = 176\,000$ байтов $= 1\,408\,000$ битов $= 1375$ Кбит.

Таким образом, формат MP3 обеспечивает степень сжатия 1375/256 ≈ 5,4 при сохранении качества звучания для человека (хотя часть информации, строго говоря, теряется).

Для сжатия видео может использоваться **алгоритм MJPEG** (англ. *Motion JPEG* — JPEG в движении), когда каждый кадр сжимается по алгоритму JPEG. Кроме того, широко применяют стандарт **MPEG-4**, разработанный специально для сжатия цифрового звука и видео.

Программы (и устройства), которые выполняют кодирование и декодирование звука и видео, называют **кодеками** (англ. *codec — coder/decoder* — кодировщик/декодировщик). Наиболее известные аудиокодеки (программы для преобразования звука) – *MP3*, *Ogg Vorbis* (*OGG*) и *AAC*. Среди видеокодеков чаще всего используются кодеки стандарта MPEG-4 (*DivX*, *Xvid*, *H.264*, *QuickTime*) и кодек *WMV* (Windows Media Video). Самые популярные свободные (англ. *open source*) и бесплатные (англ. *freeware*) кодеки собраны в пакет *K-Lite Codec Pack* (codecguide.com).

**Выводы**

Мы рассмотрели различные алгоритмы сжатия информации. Все они основаны на том, что в информации есть некоторая *избыточность*, закономерность, которую можно использовать для уменьшения объёма данных. **Хорошо сжимается** информация, в которой большая избыточность:

- тексты, в которых повторяются одинаковые слова и символы имеют разные частоты встречаемости (файлы с расширениями **txt**, **doc**, **docx**);
- документы — тексты с оформлением и, возможно, вставленными рисунками, таблицами и т. п.; электронные таблицы (файлы с расширениями **doc**, **docx**, **xls**);
- рисунки, имеющие большие области одного цвета и записанные без сжатия (файлы **bmp**);
- несжатый звук (файлы **wav**);
- несжатое видео (файлы **avi**)[1].

**Плохо сжимаются** данные, где избыточность маленькая или ее совсем нет:

[1] Файлы с расширением `avi` могут хранить как сжатое, так и несжатое видео.

- архивы, упакованные со сжатием (**zip**, **rar**, **7z** и др.);
- сжатые рисунки (файлы **gif**, **jpg**, **png**, **tif** и др.);
- сжатый звук (файлы **mp3**, **wma**);
- сжатое видео (файлы **mpg**, **wmv**, **mp4**);
- программы (файлы **exe**).

Данные невозможно сжать, если в них нет никаких закономерностей. Поэтому хуже всего сжимаются случайные числа, например полученные на компьютере. Современным программам-упаковщикам иногда удаётся их немного сжать, но не более, чем на 1–2%. Это происходит потому, что в последовательности псевдослучайных чисел, которые выдает компьютерная программа-генератор, всё же можно выявить какие-то закономерности.

Заметим, что не всегда нужно стремиться к полному устранению избыточности кода. Как вы знаете из предыдущего параграфа, именно избыточность позволяет обнаруживать и исправлять ошибки при передаче данных.

## Вопросы и задания

1. За счёт чего удаётся сжать данные без потерь? Когда это сделать принципиально невозможно?
2. Какие типы файлов сжимаются хорошо, а какие — плохо? Почему?
3. Текстовый файл, записанный в однобайтной кодировке, содержит только 33 заглавные русские буквы, цифры и пробел. Ответьте на следующие вопросы:
   - какое минимальное число битов нужно выделить на символ при передаче, если каждый символ кодируется одинаковым числом битов?
   - сколько при этом будет занимать заголовок пакета данных?
   - при какой минимальной длине текста коэффициент сжатия будет больше 1?
4. На чём основан алгоритм сжатия RLE? Когда он работает хорошо? Когда нет смысла его использовать?
5. Что такое префиксный код?
6. В каких случаях допустимо сжатие с потерями?
7. Опишите простейшие методы сжатия рисунков с потерями. Приведите примеры.
8. На чём основан алгоритм JPEG? Почему это алгоритм сжатия с потерями?
9. Что такое артефакты?
10. Для каких типов изображений эффективно сжатие JPEG? Когда его не стоит применять?

11. На чём основано сжатие звука в алгоритме MP3?
12. Что такое битрейт? Как он связан с качеством звука?
13. Какое качество звука принимается за эталон качества на непрофессиональном уровне?
14. Какие методы используются для сжатия видео?

**Подготовьте сообщение**

а) «Программы для сжатия данных»

б) «Алгоритмы сжатия изображений»

в) «Аудиокодеки»

г) «Видеокодеки»

## Задачи

1. С помощью алгоритма RLE закодируйте сообщение «ВАААААВАААРРРРРРРРРР».
2. После кодирования методом RLE получилась следующая последовательность байтов (первый байт — управляющий):

   10000011 10101010 00000010 10101111 11111111 10000101 10101010

   Сколько байтов будет содержать данная последовательность после распаковки?
3. После кодирования методом RLE получилась следующая последовательность байтов (первый байт — управляющий):

   00000011 10101010 00000010 10101111 10001111 11111111

   Сколько байтов будет содержать данная последовательность после распаковки?
4. Раскодируйте сообщение, которое закодировано с помощью приведённого в тексте кода Шеннона–Фано:
   11111000011011111001001101111001.
5. Постройте дерево, соответствующее коду А — 0, Б — 1, В — 00, Г — 01, Д — 10, Е — 11. Является ли этот код префиксным? Как это определить, посмотрев на дерево?

*6. Постройте дерево Хаффмана для фразы «МАМА МЫЛА ЛАМУ». Найдите коды всех входящих в нее символов и закодируйте сообщение. Чему равен коэффициент сжатия в сравнении с равномерным кодом минимальной длины? С однобайтной кодировкой?