Глава 2
КОДИРОВАНИЕ ИНФОРМАЦИИ

§ 4
Дискретное кодирование

Ключевые слова:

- знаковая система
- знак
- символ
- аналоговый сигнал
- цифровой сигнал
- дискретизация

Кодирование — это представление информации в форме, удобной для её хранения, передачи и автоматической обработки.
Код — это правило, по которому сообщение преобразуется в цепочку знаков.
Язык — это система знаков и правил, используемая для записи и передачи информации.
Формальный язык — это язык, в котором однозначно определяется значение каждого слова, а также правила построения предложений и придания им смысла.

Знаковые системы

Для хранения, передачи и обработки информации нужно как-то зафиксировать её, записать с помощью некоторой системы знаков (знаковой системы).

Знак — это заменитель какого-то объекта, который вызывает в сознании человека образ этого объекта. Например, знак  напоминает горнолыжника, такой знак называется *пиктограммой*.

Символ — это знак, о значении которого люди договорились, например, «§» — это символ параграфа, а ₽ — символ рубля. Если значение символов неизвестно, мы не сможем понять смысл сообщения, записанного с их помощью. Поэтому многие древние тексты до сих пор не расшифрованы.

Знаки бывают зрительные (буквы, цифры, ноты, дорожные знаки), слуховые (звуки устной речи, звуковые сигналы), осяза-

тельные (азбука Брайля для слепых людей), обонятельные (их используют животные).

Знаковая система определяется **алфавитом** (набором используемых знаков) и правилами выполнения операций с этими знаками.

Знакомые всем знаковые системы — это **естественные языки**, с помощью которых люди общаются в быту (русский, английский, китайский и др.). В естественных языках кроме правил есть и исключения. Кроме того, одно и то же слово может иметь различный смысл в зависимости от *контекста*, т. е. отрывка текста, в котором оно употребляется.

В научных публикациях такая ситуация недопустима, потому что смысл текста должен быть понят однозначно. В таких случаях используют **формальные языки**, в которых каждое слово и словосочетание имеют чётко определённое единственное значение, и нет никаких исключений. Примеры формальных языков — математические и химические формулы, нотная запись, языки программирования. Все формальные языки — *искусственные*, они разработаны людьми для обмена информацией в специальных областях знаний.

Как вы знаете из курса основной школы, в компьютерах для кодирования всех видов информации используется специальная знаковая система — **двоичный код**. Давайте разберёмся, почему такое решение оказалось самым лучшим.

Аналоговые и дискретные сигналы

Как вы уже знаете, информация передаётся в закодированном виде с помощью сигналов. Согласно определению, сигнал — это изменение свойств носителя, которое используется для передачи информации. Изменение выбранного свойства (например, силы тока, напряжения, освещённости) во времени можно описать в виде функции. Далее такую функцию тоже будем называть сигналом, как это принято в электронике и вычислительной технике.

В любой компьютерной системе очень важно обеспечить надёжный обмен данными в условиях, когда на сигнал действуют помехи. Поэтому желательно выбрать такой способ кодирования информации, который позволяет лучше всего решить эту задачу.

Элементы электронных устройств, как правило, обмениваются данными с помощью электрических сигналов; для получения информации приёмник должен измерить этот сигнал (чаще всего — напряжение на контактах или силу тока). В таких устройствах,

как радиоприёмник и микрофон, изменение электрического сигнала может произойти в любой момент и быть любым (в пределах допустимого диапазона). Такие сигналы называют аналоговыми.

Аналоговый сигнал — это сигнал, который в любой момент времени может принимать любые значения в заданном диапазоне.

Органы чувств человека воспринимают информацию в аналоговой форме: свет, звуковые волны, вкус, запах и т. п. Поэтому раньше большинство технических устройств для работы с информацией (телефоны, магнитофоны, фотоаппараты) тоже были аналоговыми.

В 60-х годах XX века были широко распространены *аналоговые компьютеры*, которые выполняли вычисления с аналоговыми сигналами (сложение, вычитание, умножение, извлечение квадратного корня). Однако они решали достаточно узкий круг задач (моделирование законов движения), и их точность была невысока.

Дело в том, что при передаче сигнала всегда есть помехи, которые искажают его значения. В большинстве случаев эти искажения — случайные ошибки, не поддающиеся учёту. Фактически приёмник получает не исходный сигнал, посланный источником (сплошная линия на рис. 2.1), а искажённый (штриховая линия).

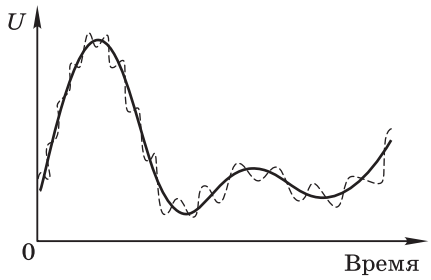


Рис. 2.1

Вспомним, что аналоговый сигнал может принимать любые значения в некотором диапазоне. «Очистить» его от помех в общем случае нельзя, потому что невозможно понять, искажён он или на самом деле имеет такое значение. Кроме того, дополнительные ошибки (погрешности) вносятся при измерении сигнала.

Если использовать аналоговые компьютеры, мы будем при каждом расчёте с одинаковыми исходными данными получать несколько отличающиеся результаты. Кроме того, при копировании аналоговая информация искажается (например, при каждом

копировании звукозаписи на магнитной ленте качество копии ухудшается).

Эта ситуация не устраивала инженеров, разрабатывающих компьютеры, и они нашли интересное решение: если не удаётся точно измерить сигнал, нужно вообще отказаться от его измерения, а просто через некоторый интервал времени T определять, в каком из двух состояний находится сигнал (эти состояния можно обозначить как 1 и 0)[1]. Тогда мы получаем огромное преимущество: при небольших помехах искажение сигнала не влияет на передачу данных: если напряжение выше некоторого порога U_1, то считается, что сигнал равен 1, а все сигналы, меньшие другого порога U_0, считаются равными нулю (рис. 2.2).

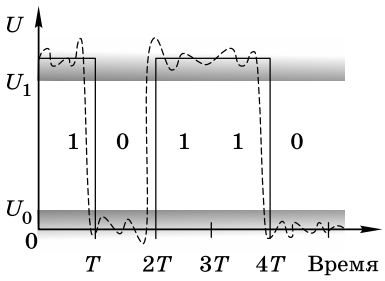


Рис. 2.2

Сигналы, с которыми работает компьютер, называются **дискретными** или **цифровыми**. Они обладают двумя важными свойствами:

- изменяются только в отдельные моменты времени (*дискретность по времени*);
- принимают только несколько возможных значений (*дискретность по уровню*).

Дискретный (цифровой) сигнал — это последовательность значений, принадлежащих некоторому конечному множеству.

Обратите внимание на важный момент — мы естественным образом пришли к необходимости использования дискретных сигналов, когда потребовалось точно и однозначно воспринимать передаваемую информацию с учётом неизбежных помех.

Так как каждому значению дискретного сигнала всегда можно поставить в соответствие определённый знак, такой сигнал можно рассматривать как сообщение, записанное с помощью конечного набора знаков (алфавита).

[1] Может быть и наоборот: 0 обозначает, что сигнал есть, а 1 — сигнала нет.

Этот принцип применим не только к компьютерам. Переход от наскальных рисунков к алфавитному письму, где каждый знак имеет чётко определённое значение, — это переход от аналоговых сигналов к дискретным, цель которого — максимально исключить неоднозначное понимание смысла. Код Морзе и двоичный код — это тоже дискретные коды.

Дискретизация

Поскольку данные в компьютерах передаются с помощью дискретных сигналов, они могут хранить и обрабатывать только **дискретную информацию**, т. е. такую, которая может быть записана с помощью конечного количества знаков некоторого алфавита. Поэтому для ввода любых данных в компьютер их нужно перевести в дискретный код. Например, линия, нарисованная на бумаге, при сканировании представляется в памяти компьютера в виде отдельных элементов — пикселей. Такая процедура называется дискретизацией.

Дискретизация означает, что мы представляем целое (непрерывное) в виде набора отдельных элементов. Например, картина художника — это аналоговая (непрерывная) информация, а мозаика, сделанная на её основе (рисунок из кусочков разноцветного стекла), — дискретная.

Дискретизация — это представление непрерывного объекта в виде множества отдельных элементов.

Дискретизацию мы используем и в жизни. Например, когда измеряют температуру воздуха, обычно округляют её до целых градусов, хотя температура изменяется непрерывно, а не скачками: она может быть равной и 18,25 °C, и 18,251 °C, и 18,2513 °C и т. д.

Множество вещественных чисел непрерывно (между любыми двумя различными числами есть ещё бесконечно много других), а множество целых чисел — дискретно. Таким образом, при округлении мы выполняем дискретизацию данных.

Все приборы, которые показывают результаты измерений в цифровом виде, выполняют дискретизацию. Например, стрелка в обычном спидометре автомобиля может принимать любое положение, это непрерывный (*аналоговый*) прибор. А цифровой спидометр показывает дискретные данные — скорость с округлением до 1 км/ч.

Всем известное иррациональное число π содержит бесконечное количество знаков в дробной части. Если мы хотим записать, чему равно π, необходимо остановиться на каком-то знаке, отбросив остальные, например $\pi \approx 3{,}14$. Таким образом, мы перешли к дискретной информации, потому что рассматриваем только числа с шагом 0,01 — точки на числовой оси (рис. 2.3).

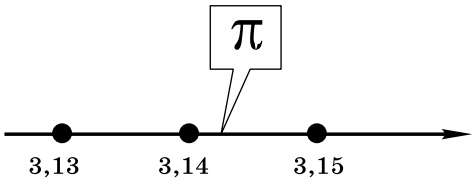
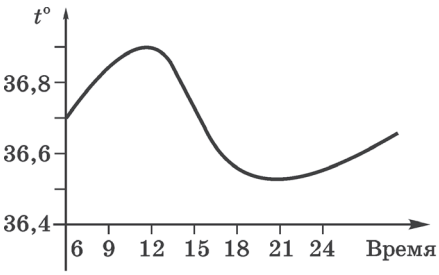
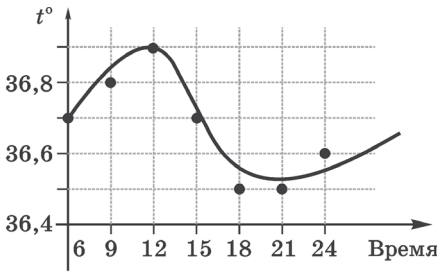


Рис. 2.3

Изменение высоты столбика термометра — это аналоговые данные, а *записанная* температура, округлённая до десятых долей градуса (например, 36,6°), — дискретные. Дискретность состоит в том, что записанные значения температуры изменяются скачкообразно (через 0,1°), — это дискретизация по уровню, или *квантование*. Кроме того, обычно температуру больного измеряют не непрерывно, а несколько раз в день — появляется дискретизация по времени (рис. 2.4).



Аналоговые данные



Дискретизация

6 ч. 36,7°
9 ч. 36,8°
12 ч. 36,9°
15 ч. 36,7°
18 ч. 36,5°
21 ч. 36,5°
24 ч. 36,6°

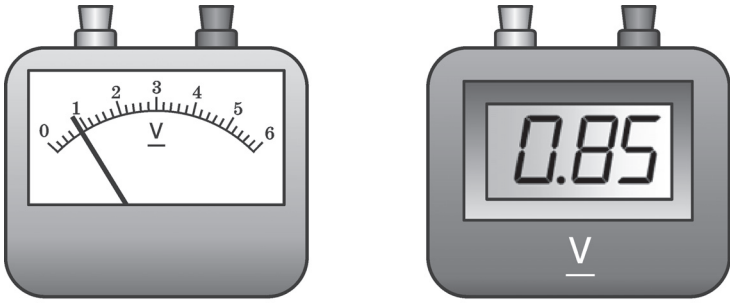
Дискретные данные

Рис. 2.4

Заметим, что при дискретизации, как правило, происходит *потеря информации*. В данном случае мы, во-первых, потеряли информацию об изменении температуры между моментами измерений и, во-вторых, исказили измеренные значения, округлив их до десятых (каждая дискретизация, и по времени, и по уровню, вносит свою ошибку). Чтобы уменьшить ошибки, нужно уменьшать шаг дискретизации — измерять температуру чаще, записывать показания термометра до тысячных долей градуса и т. д. Однако в любой практической задаче есть некоторый предел, после которого увеличение точности уже никак не влияет на конечный результат.

Из приведённого примера понятно, что непрерывность и дискретность — это не свойство самой информации, а свойство её *представления*. В данном случае информация — это сведения об изменении температуры человека в течение дня. Если бы она измерялась постоянно и записывалась самописцем (в виде графика), можно было бы говорить о том, что эта информация представлена в аналоговой (непрерывной) форме.

Ещё один пример — аналоговые («стрелочные») и цифровые вольтметры, которые измеряют одну и ту же величину, но выводят результат измерения в разном виде (рис. 2.5).



Аналоговый прибор

Дискретный прибор

Рис. 2.5

С одной стороны, переход к дискретному представлению информации делает более надёжной передачу данных (если обе стороны одинаково понимают используемые знаки). С другой стороны, при дискретизации часть информации теряется.

Хотя аналоговую информацию невозможно точно представить в дискретном виде, при увеличении точности дискретизации свойства непрерывной и дискретной информации практически совпадают. Например, для точной записи числа π требуется бесконеч-

ное количество цифр, но в расчётах чаще всего достаточно знать это значение с точностью не более 10 знаков.

Идеальная непрерывность существует только в теории. Мы считаем дерево, пластмассу, металл непрерывными, но на самом деле они состоят из отдельных молекул, расположенных на некотором расстоянии друг от друга, — это значит, что вещество дискретно.

Иллюстрация в книге кажется нам сплошной, но при сильном увеличении видно, что она строится из отдельных точек (имеет «растр», рис. 2.6). Классическая («плёночная») фотография считается аналоговой, но при увеличении снимка с фотоплёнки нельзя бесконечно получать всё новые и новые детали — предел «уточнения» определяется величиной зерна светочувствительного материала.

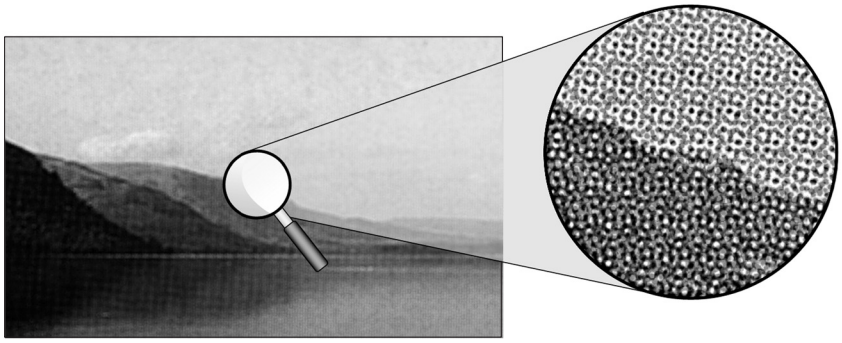


Рис. 2.6

Мы часто воспринимаем дискретные объекты как непрерывные, потому что наши органы чувств не позволяют различить отдельные элементы. Например, *разрешающая способность* глаза составляет около одной угловой минуты ($1' = 1/60$ часть градуса), это значение определяется размером элементов сетчатки глаза. Поэтому человек не может различить два объекта, если направления на них отличаются меньше, чем на $1'$. Для того чтобы повысить разрешающую способность при наблюдении, применяют специальные приборы (например, бинокли и микроскопы).

Выводы

- Знак — это заменитель какого-то объекта, который вызывает в сознании человека образ этого объекта.
- Знаковая система определяется алфавитом (набором используемых знаков) и правилами выполнения операций с этими знаками.

- Формальный язык — это язык, в котором однозначно определяется значение каждого слова, а также правила построения предложений и придания им смысла.
- Аналоговый сигнал — это сигнал, который в любой момент времени может принимать любые значения в заданном диапазоне.
- Дискретный (цифровой) сигнал — это последовательность значений, каждое из которых принадлежит некоторому конечному множеству.
- Дискретизация — это представление непрерывного объекта в виде множества отдельных элементов.

Интеллект-карта

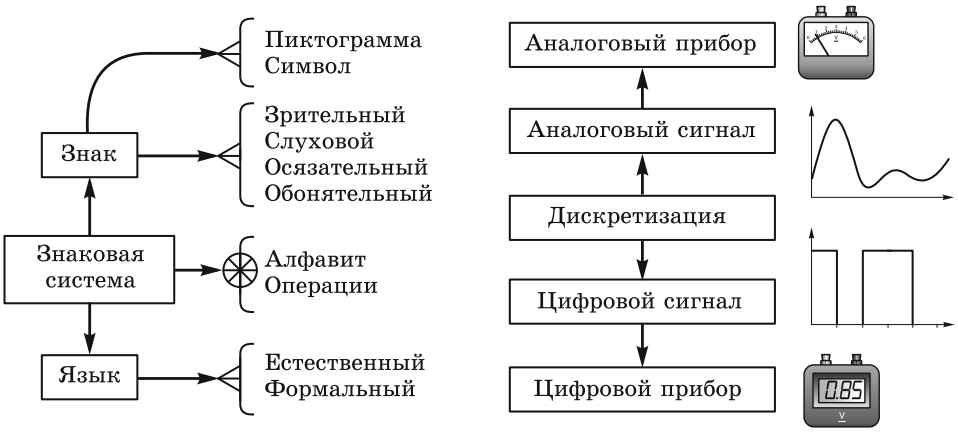


Рис. 2.7

Вопросы и задания

1. В чём различие между пиктограммой и символом?
2. Приведите примеры знаковых систем, которые не упоминаются в учебнике.
3. Какие бытовые устройства работают с аналоговыми сигналами?
4. Какие системы связи используют аналоговые сигналы, а какие — дискретные?
5. Почему при использовании аналоговой техники передача информации всегда происходит с искажениями?
6. Почему с помощью дискретного сигнала можно передавать информацию практически без искажений? Искажается ли форма такого сигнала под воздействием помех?

7. Сигнал изменяется в моменты времени, кратные 1 секунде, и может принимать одно из 16 возможных значений. Можно ли назвать такой сигнал дискретным?
8. Что такое дискретизация по времени и дискретизация по уровню?
9. Приведите пример дискретизации сигнала: а) только по уровню; б) только по времени. Нарисуйте графики процессов.
10. Почему при дискретизации, как правило, происходит потеря информации? В каких случаях потери информации не будет?
11. Как можно уменьшить потери информации при дискретизации? Почему не стоит стремиться максимально уменьшить эти потери?
12. Приведите примеры, когда одна и та же информация может быть представлена в аналоговой и дискретной формах.

*13. Выясните, какие музыкальные инструменты позволяют извлекать только дискретные звуки (заранее определённые ноты), а какие — звук любой частоты.

14. Объясните фразу «при увеличении точности дискретизации свойства непрерывной и дискретной информации практически совпадают».

Проекты

а) Аналоговые и дискретные музыкальные инструменты
б) Дискретность в полиграфии

§ 5
Равномерное и неравномерное кодирование

Ключевые слова:

- кодирование
- равномерный код
- неравномерный код
- правило умножения
- правило сложения

Алфавит — это набор знаков, который используется в языке.
Мощность алфавита — это количество знаков в алфавите.
Равномерный код — это код, в котором все кодовые слова имеют одинаковую длину.
Неравномерный код — это код, в котором кодовые слова имеют различную длину.
Двоичное кодирование — это кодирование с помощью двух знаков.
1 бит — это одна двоичная цифра (один знак сообщения, записанного в двоичном коде).

Равномерное кодирование

Если алфавит языка состоит из M знаков (имеет мощность M), то количество различных сообщений длиной L знаков вычисляется как

$$N = M^L.$$

Двоичный код — это сообщение, использующее алфавит из двух знаков (0 и 1), поэтому для двоичного кода эта формула запишется в виде

$$N = 2^L.$$

Например, восьмиразрядная ячейка памяти компьютера может хранить одно из $2^8 = 256$ различных значений.

Если заданное количество вариантов не равно степени числа 2, выбирают длину кода с запасом. Например, для кодирования номера спортсмена в интервале от 1 до 200 нужно использовать не меньше, чем 8 двоичных разрядов (бит), поскольку 7 бит позволяют закодировать только 128 различных значений (этого мало!), а 8 бит — уже 256:

$$2^7 = 128 < 200 \le 256 = 2^8.$$

В середине XX века в СССР была разработана электронно-вычислительная машина «Сетунь», в которой для хранения и обработки данных использовался троичный код с алфавитом {–1, 0, 1}. В таком компьютере восьмиразрядная ячейка памяти могла хранить одно из $3^8 = 6561$ значения.

Правило умножения

При дополнительных ограничениях количество возможных знаков в разных позициях сообщения может различаться, поэтому нужно использовать более общее **правило умножения:**

$$N = M_1 \cdot M_2 \cdot \ldots \cdot M_L.$$

Здесь M_k — это возможное количество вариантов выбора знака в позиции k.

Задача 1. Сколько существует различных сообщений длины 5 в четырёхбуквенном алфавите {A, B, C, X}, если известно, что буква «X» может появляться только на первом или на последнем месте?

Решение. По условию на первом и последнем местах может быть любая из четырёх букв, поэтому $M_1 = M_5 = 4$. В остальных трёх позициях могут быть любые буквы, кроме «X», поэтому $M_2 = M_3 = M_4 = 3$. Тогда общее количество различных сообщений равно $N = 4 \cdot 3 \cdot 3 \cdot 3 \cdot 4 = 432$.

Ответ: 432.

Задача 2. Сколько существует пятизначных десятичных чисел, в которых каждая цифра встречается только один раз?

Решение. Представим себе, что нам нужно заполнить пять ячеек разными цифрами. Так как у нас есть 10 цифр (от 0 до 9), первую ячейку мы можем заполнить 10 способами, ведь остальные ячейки ещё пустые и никаких ограничений нет. Заполняя вторую ячейку, мы уже не можем использовать одну цифру, которую выбрали для первой ячейки, поэтому остаётся только 9 вариантов. При заполнении третьей ячейки нужно учитывать, что для первых двух цифры уже выбраны и их использовать нельзя, остаётся 8 свободных цифр, и т. д. (рис. 2.8).

Рис. 2.8

Ответ: 30 240 чисел.

Задача 3. Вася составляет 5-буквенные слова, в которых есть только буквы «К», «Р», «О», «Т», причём буква «О» используется в каждом слове ровно 1 раз. Каждая из других допустимых букв может встречаться в слове любое количество раз или не встречаться совсем. Словом считается любая допустимая последовательность букв, не обязательно осмысленная. Сколько существует таких слов, которые может написать Вася?

Решение. Единственная буква «О» может стоять в любой из 5 позиций. Предположим, что мы определили её место, тогда на каждом из оставшихся 4 мест может оказаться любая из остальных трёх букв («К», «Р» или «Т). Таким образом, при любой заданной позиции буквы «О» Вася может составить $3^4 = 81$ слово, а всего существует $5 \cdot 3^4 = 405$ пятибуквенных слов с единственной буквой «О».

Ответ: 405.

Неравномерное кодирование

При неравномерном кодировании коды различных знаков могут иметь разную длину. Это делается для того, чтобы сократить длину сообщения, используя сведения о частотах встречаемости различных знаков. Знаки, которые встречаются в сообщениях чаще других (например, для текстов на русском языке — это пробел и буквы «О», «Е» и «А») получают более короткие коды, а редко встречающиеся знаки — более длинные.

Задача 4. По каналу связи передаются сообщения, каждое из которых содержит 16 букв «А», 8 букв «Б», 4 буквы «В» и 4 буквы «Г» (других букв в сообщениях нет). Каждую букву кодируют двоичной последовательностью. Какой код из приведённых ниже следует выбрать, чтобы длина сообщения была как можно меньше?

код 1: А — 0, Б — 10, В — 110, Г — 111;
код 2: А — 00, Б — 01, В —10, Г — 11.

Решение. Считаем общее количество бит в сообщении для кода 1:

$$16 \cdot 1 + 8 \cdot 2 + 4 \cdot 3 + 4 \cdot 3 = 56 \text{ бит}$$

и общее количество бит в сообщении для кода 2:

$$16 \cdot 2 + 8 \cdot 2 + 4 \cdot 2 + 4 \cdot 2 = 64 \text{ бит}.$$

Код 1 даёт наименьшую длину сообщения, поэтому выбираем его.

Ответ: код 1.

Задача 5. По каналу связи передаются сообщения, содержащие только 4 буквы: «А», «И», «С», «Т». В любом сообщении больше всего букв «А», следующая по частоте буква — «С», затем — «И». Буква «Т» встречается реже, чем любая другая. Для передачи сообщений нужно использовать один из следующих неравномерных двоичных кодов:

код 1: А — 1, И — 01, С — 001, Т — 000;
код 2: А — 101, И — 11, С — 0, Т — 100

так, чтобы сообщения были как можно короче. Какой код следует выбрать?

Решение. Для того чтобы длина сообщения была наименьшей, должно выполняться правило: «чем чаще встречается буква, тем короче её код». К сожалению, это правило не выполняется для кодов 1 и 2: в коде 1 длина кодового слова для буквы «С» боль-

ше, чем длина кодового слова для буквы «И» (а лучше было бы сделать наоборот!); для кода 2 длина кодового слова для буквы «А» — не самая маленькая из всех.

Предположим, что в сообщении буква «А» встречается α раз, буква «И» — β раз, буква «С» — γ раз и буква «Т» — δ раз, причём по условию задачи $\alpha > \gamma > \beta > \delta$. При кодировании с помощью кода 1 получаем сообщение длиной

$$S_1 = \alpha + 2\beta + 3\gamma + 3\delta,$$

а при кодировании с помощью кода 2 длина сообщения равна

$$S_2 = 3\alpha + 2\beta + \gamma + 3\delta.$$

Находим разность: $S_2 - S_1 = (3\alpha + 2\beta + \gamma + 3\delta) - (\alpha + 2\beta + 3\gamma + 3\delta) = 2\alpha - 2\gamma$. Поскольку $\alpha > \gamma$, получаем: $S_2 - S_1 > 0$, т. е. код 1 более экономичный.

Ответ: код 1.

Задача 6. Сколько символов можно закодировать с помощью кода Морзе, используя кодовые слова различной длины — от 1 до 5 знаков?

Решение. Кодовые слова длиной, скажем, 2 знака можно выбирать независимо от кодовых слов другого размера. Поэтому достаточно найти количество кодовых слов N_L для каждого возможного значения длины кодового слова L и сложить эти числа:

$$N = N_1 + N_2 + N_3 + N_4 + N_5.$$

Это правило называется **правилом сложения**.

Так как в коде Морзе используются всего два знака (точка и тире), из общей формулы получаем: $N_L = 2^L$. Поэтому $N = 2^1 + 2^2 + 2^3 + 2^4 + 2^5 = 62$.

Ответ: 62.

Выводы

- Кодирование — это представление информации в форме, удобной для её хранения, передачи и обработки. Правило такого преобразования называется кодом. Кодом называют также набор знаков закодированного сообщения.
- Если алфавит языка состоит из M символов (имеет мощность M), количество различных сообщений длиной L знаков вычисляется как $N = M^L$. Для двоичного кода эта формула запишется в виде $N = 2^L$.

- Правило умножения: $N = M_1 \cdot M_2 \cdot \ldots \cdot M_L$, где M_k — это возможное количество вариантов выбора знака в позиции k сообщения.
- Правило сложения: $N = N_k + N_{k+1} + \ldots + N_m$, где N_q ($q = k, \ldots, m$) — это количество различных сообщений длиной q.

Интеллект-карта

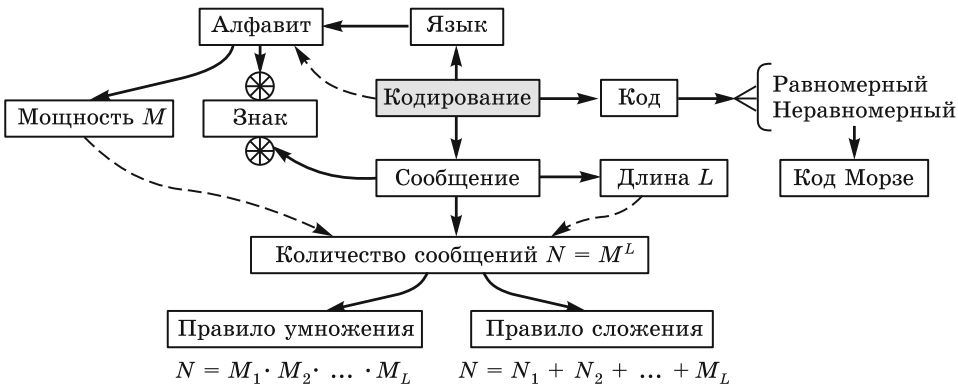


Рис. 2.9

Подготовьте сообщение

«Основные формулы комбинаторики»

Проекты

а) Оптимальный код Морзе для русского языка
б) Программа для частотного анализа текстов

§ 6 Декодирование

Ключевые слова:

- декодирование
- условие Фано
- префиксный код
- постфиксный код

Декодирование — это восстановление информационного сообщения из последовательности кодов.

Сообщения, записанные с помощью равномерного кода, всегда декодируются однозначно. Для этого достаточно разбить сообщение на группы известной длины и декодировать каждую группу по кодовой таблице.

Условие Фано

В некоторых случаях даже при использовании неравномерного кода не требуется вводить символ-разделитель. Для этого достаточно выполнения условия Фано: ни одно кодовое слово не является началом другого кодового слова. Такой код называют **префиксным**.

Пример 1. Пусть для кодирования первых пяти букв русского алфавита используется кодовая таблица:

А	Б	В	Г	Д
000	10	01	110	001

Это неравномерный код, поскольку в нём есть двух- и трёхзначные коды. Построим для этой кодовой таблицы дерево, в котором от каждого узла (кроме листьев) отходят два ребра, помеченные цифрами 0 и 1. Чтобы найти код символа, нужно пройти по стрелкам от корня дерева к нужному листу, выписывая метки стрелок, по которым мы переходим (рис. 2.10).

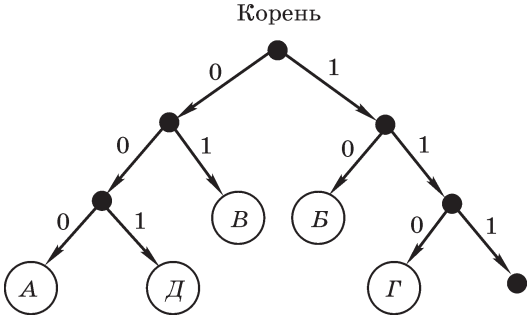


Рис. 2.10

Заметим, что ни одна буква не лежит на пути от корня к другой букве. Это значит, что условие Фано выполняется, и любую правильную кодовую последовательность можно однозначно декодировать с начала. Например, рассмотрим цепочку 1100000100110. Букв с кодами 1 и 11 в таблице нет, поэтому сообщение начинается с буквы «Г» — она имеет код 110:

Г
110 ¦ 0000100110

Следующий (единственно возможный) код — 000, это буква «А»:

Г	А	
110 ¦	000 ¦	0100110

Аналогично декодируем все сообщение:

Г	А	В	Д	Б
110 ¦	000 ¦	01 ¦	001 ¦	10

Пример 2. Рассмотрим другую кодовую таблицу:

А	Б	В	Г	Д
000	01	10	011	100

Здесь условие Фано не выполняется, поскольку код буквы «Б» (01) является началом кода буквы «Г» (011), а код буквы «Д» (100) начинается с кода буквы «В» (10). Дерево для этой кодовой таблицы выглядит так (рис. 2.11).

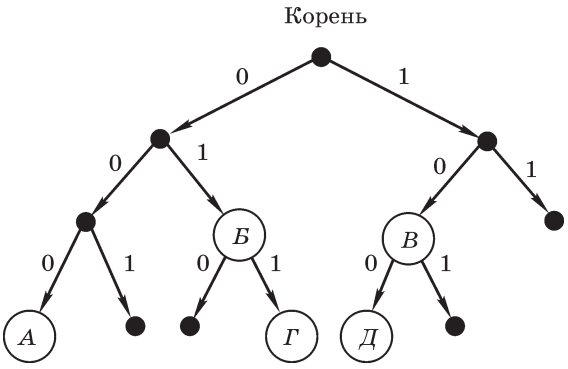


Рис. 2.11

Тем не менее можно заметить, что выполнено «обратное» условие Фано: ни одно кодовое слово не является окончанием другого кодового слова (такой код называют **постфиксным**). Поэтому закодированное сообщение можно однозначно декодировать с конца. Например, рассмотрим цепочку 011000110110. Последней буквой в этом сообщении может быть только «В» (код 10):

	В
0110001101 ¦	10

Вторая буква с конца — «Б» (код 01):

	Б	В
01100011 ¦	01 ¦	10

и так далее:

Б	Д	Г	Б	В
01	100	011	01	10

В общем случае декодировать сообщение удаётся только перебором вариантов.

Пример 3. Декодируем сообщение 010100111101, закодированное с помощью кодовой таблицы:

А	Б	В	Г	Д
01	010	011	11	101

Здесь не выполняется ни «прямое», ни «обратное» условие Фано, поэтому декодировать сообщение однозначно, возможно, не удастся. На первом месте может быть буква «А» или буква «Б». Сначала предположим, что это буква «А»:

А0100111101

Тогда второй буквой также может быть буква «А»:

АА00111101.

Дальше декодировать не получается, потому что в таблице нет кодов 0, 00 и 001. Поэтому проверяем второй вариант — вторая буква — «Б»:

АБ0111101.

Третьей буквой может быть «А»:

АБА11101,

Тогда четвёртая и пятая буквы определяются однозначно — это буквы «Г» и «Д». Таким образом, один из подходящих вариантов — АБАГД.

Посмотрим, есть ли другие варианты. После сочетания АБ может стоять буква «В»:

АБВ1101,

тогда оставшиеся буквы — это ГА, а полное сообщение — АБВГА. Этот вариант тоже подходит.

Кроме того, на первом месте может стоять буква «Б»:

Б100111101,

но дальше декодировать не удаётся, потому что в таблице нет кодов 1, 10 и 100. Таким образом, сообщение может быть декодировано двумя способами: АБАГД и АБВГА.

Пример 4. Для кодирования некоторой последовательности, состоящей из букв «А», «Б», «В» и «Г», решили использовать неравномерный двоичный код, удовлетворяющий условию Фано. Для буквы «А» использовали кодовое слово 0, для буквы «Б» — кодовое слово 110. Требуется найти коды для букв «В» и «Г», при которых суммарная длина всех четырёх кодовых слов минимальна.

Построим дерево, которое содержит известные кодовые слова для букв «А» и «Б» (рис. 2.12).

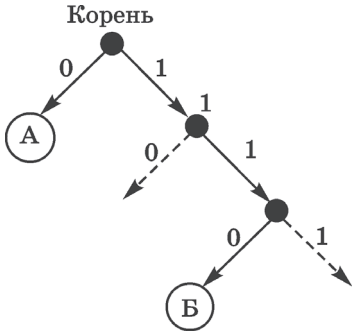


Рис. 2.12

Штриховыми линиями показаны ветви дерева, на которых нужно разместить коды оставшихся букв. Поскольку требуется обеспечить выполнение условия Фано, кодовые слова для них должны соответствовать листьям этого дерева. В данном случае есть две свободных ветви, и нужно выбрать два кодовых слова (для букв «В» и «Г»). Поэтому можно использовать кодовые слова 10 и 111, так что суммарная длина всех четырёх кодовых слов будет равна 1 + 3 + 2 + 3 = 9 (рис. 2.13).

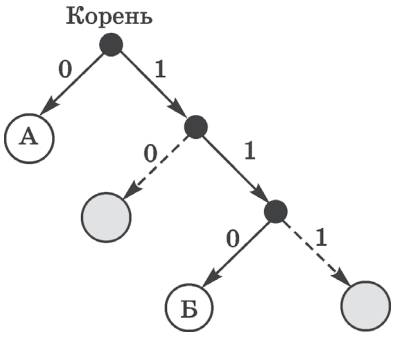


Рис. 2.13

Если бы требовалось выбрать не два, а три кодовых слова (скажем, для букв «В», «Г» и «Д»), на одной из ветвей (самой короткой!) нужно было бы сделать развилку (рис. 2.14).

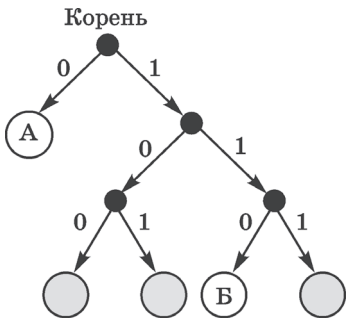


Рис. 2.14

Листья дерева, которые соответствуют оптимальному выбору трёх кодовых слов, выделены фоном на рис. 2.14.

Граф Ал. А. Маркова

Пример 3 показывает, что неоднозначное декодирование возможно тогда, когда начало кода одной буквы совпадает с концом кода другой, поэтому можно переместить границу между кодами букв в сообщении. Например, последовательность 01011 может быть декодирована как АВ (01011) и как БГ (01011). Следовательно, нужно обратить внимание на те цепочки, которые встречаются как в начале, так и в конце кодовых слов.

Покажем, как найти сообщения, которые декодируются неоднозначно. Для таблицы из примера 3 построим граф Ал. А. Маркова следующим образом:

1. Определим все последовательности, которые совпадают с началом какого-то кодового слова и одновременно с концом какого-то кодового слова; в данном случае это три последовательности:
   0 (начало кода буквы «А» и конец кода буквы «Б»);
   1 (начало кода буквы «Г» и конец кода буквы «Д»);
   10 (начало кода буквы «Д» и конец кода буквы «Б»).
   Цепочки 01 и 11 не учитываем, потому что они совпадают с кодами букв «А» и «Г».
2. Добавим к этому множеству {0, 1, 10} пустую строку, которую обычно обозначают буквой Λ (прописная гречес-

кая буква «лямбда»); элементы полученного множества {Λ, 0, 1, 10} становятся вершинами графа (рис. 2.15).

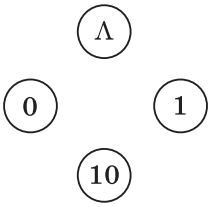


Рис. 2.15

3. Соединим вершины дугами (направленными рёбрами) по такому правилу: две вершины X и Y соединяются дугой, если последовательная запись кода вершины X, кода некоторой буквы (или нескольких букв) и кода вершины Y даёт код ещё одной буквы (рис. 2.16).

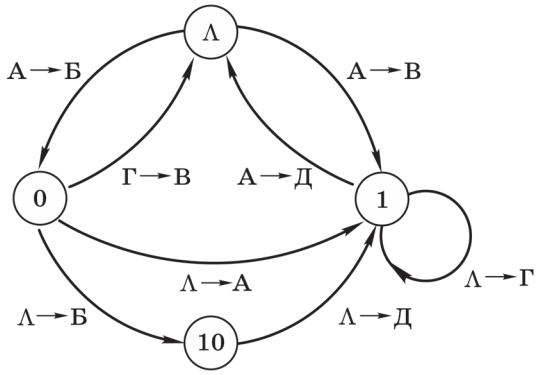


Рис. 2.16

Например, последовательная запись пустой строки (Λ), кода буквы «А» (01) и цепочки 0 даёт цепочку 010, которая совпадает с кодом буквы «Б»; поэтому рисуем дугу из вершины Λ в вершину 0; у этой дуги пишем «А → Б» и т. д. Поскольку код буквы «Г» можно записать как 11 = 1Λ1, у вершины 1 появляется петля «Λ → Г».

Любое сообщение декодируется однозначно тогда и только тогда, когда в полученном таким образом графе нет циклов, включающих вершину Λ.

В нашем графе есть несколько таких циклов, например:

- цикл Λ0Λ, соответствующий сообщению ΛА0ГΛ = 01011; это сообщение может быть расшифровано как АВ и как БГ;
- цикл Λ1Λ, соответствующий сообщению ΛА1АΛ = 01101; это сообщение может быть расшифровано как АД и как ВА;
- цикл Λ01Λ, соответствующий сообщению ΛА01АΛ = 010101; это сообщение может быть расшифровано как ААА и как БД.
- цикл Λ0101Λ, соответствующий сообщению ΛА0101АΛ = = 01010101; это сообщение может быть расшифровано как АБД и как БДА.

Кроме того, из-за петли у вершины 1 неоднозначно декодируется любая последовательность вида 01...101, где многоточие обозначает любое количество единиц. Например, сообщение 0111101 может быть декодировано как АГД или ВГА (см. пример 3).

Пример 5. Существуют коды, для которых условия Фано не выполняются, но все сообщения однозначно декодируются. В кодовой таблице

А	Б	В
0	11	010

код буквы «А» совпадает как с началом, так и с окончанием кода буквы «В», т. е. код не является ни префиксным, ни постфиксным.

Проверим, можно ли однозначно декодировать сообщения, построенные с помощью такого кода. Множество цепочек, которые совпадают с началом и концом кодовых слов, состоит из пустой строки и единицы: {Λ, 1}. Граф, построенный с помощью приведённого выше алгоритма, содержит две вершины и одну петлю (рис. 2.17).

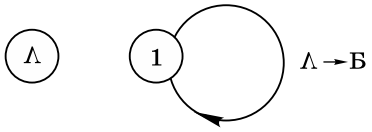


Рис. 2.17

В этом графе нет цикла, содержащего вершину Λ, поэтому любое сообщение, записанное с помощью такого кода, декодируется однозначно. Это можно показать и простыми рассуждениями:

1) все цепочки 11 в сообщении — это коды букв «Б», иначе они не могут образоваться;
2) все цепочки 010 — это коды букв «В»;
3) остальные знаки в сообщении могут быть только нулями — это коды букв «А».

Выводы

- Декодирование — это восстановление информационного сообщения из последовательности кодов.
- Сообщения, записанные с помощью равномерного кода, всегда декодируются однозначно.
- Для того чтобы сообщение, закодированное с помощью неравномерного кода, можно было однозначно декодировать, достаточно выполнения условия Фано: ни одно кодовое слово не является началом другого кодового слова. Такой код называют префиксным.
- Условие Фано выполняется тогда и только тогда, когда в дереве, построенном по кодовой таблице, все вершины-знаки являются листьями.

Нарисуйте в тетради интеллект-карту этого параграфа.

Вопросы и задания

1. Перечислите достаточные условия, при которых можно однозначно декодировать сообщение, закодированное с помощью неравномерного кода.
2. Как построить дерево для проверки выполнения обратного условия Фано?
3. В каких случаях для декодирования приходится использовать перебор вариантов?
4. *Работа в парах*. Придумайте слово из 5–6 букв и закодируйте его с помощью неравномерного кода, для которого выполняется условие Фано. Предложите напарнику декодировать сообщение.

Проекты

а) Программа для декодирования сообщений
б) Программа для проверки условия Фано

§ 7
Алфавитный подход к оценке количества информации

***Ключевые слова*:**

- алфавит
- мощность алфавита
- двоичный код
- бит

1 байт = 8 бит = 2^3 бит.
1 Кбайт (килобайт) = 1024 байта = 2^{10} байт = 2^{13} бит.
1 Мбайт (мегабайт) = 1024 Кбайт = 2^{10} Кбайт = 2^{20} байт = $= 2^{23}$ бит.
1 Гбайт (гигабайт) = 1024 Мбайт.
1 Тбайт (терабайт) = 1024 Гбайт.
1 Пбайт (петабайт) = 1024 Тбайт.

Представьте себе, что вы много раз бросаете монету и записываете результат очередного броска как 1 (если монета упала гербом) или 0 (если она упала «решкой»). В результате получится некоторое *сообщение* — цепочка нулей и единиц: 0101001101001110. Вы наверняка поняли, что здесь используется двоичное кодирование — это сообщение написано на языке, *алфавит* которого состоит из двух символов (знаков): 0 и 1. Как вы знаете, каждая двоичная цифра несёт 1 бит информации, поэтому полная информация в сообщении 0101001101001110 равна 16 бит.

Теперь представим себе, что нужно закодировать программу для Робота, который умеет выполнять команды «вперёд», «назад», «влево» и «вправо». Для этого можно использовать алфавит, состоящий из 4 символов: ↑↓→←. Сколько информации содержится в сообщении ↑←↑↑→↓↓↓↓→←? Каждый полученный символ может быть любым из 4 символов алфавита, а для кодирования одного из 4 вариантов требуется уже 2 бита. Поэтому полное сообщение из 11 символов содержит $11 \cdot 2 = 22$ бита информации.

Алфавитный подход к оценке количества информации состоит в следующем:

1) определяем мощность алфавита M (количество символов в алфавите);
2) по таблице степеней числа 2 определяем минимальное количество бит информации i, приходящихся на каждый символ сообщения, так чтобы выполнилось условие $2^i \geq M$:

$N = 2^i$	2	4	8	16	32	64	128	256	512	1024
i, бит	1	2	3	4	5	6	7	8	9	10

3) умножаем i на число символов в сообщении L, это и есть полное количество информации:

$$I = L \cdot i.$$

Обратим внимание на две важные особенности алфавитного подхода.

При использовании алфавитного подхода не учитывается, что некоторые символы могут встречаться в сообщении чаще других. Считается, что каждый символ несёт одинаковое количество информации.

Алфавитный подход не учитывает также частоты появления *сочетаний символов* (например, после гласных букв никогда не встречается мягкий знак).

Кроме того, никак не учитывается смысл сообщения, оно представляет собой просто набор знаков, которые приёмник, возможно, даже не понимает.

При использовании алфавитного подхода смысл сообщения не учитывается. Количество информации определяется только длиной сообщения и мощностью алфавита.

Во многих задачах такой подход очень удобен. Например, для устройств, передающих информацию по сети, её содержание не имеет никакого значения, важен только объём. Почтальону всё равно, что написано в письмах, важно только их количество, которое влияет на вес сумки. Для компьютера все данные — это последовательности нулей и единиц, их смысла он не понимает.

Для вычисления информационного объёма текста чаще всего применяют именно алфавитный подход.

Задача 1. Найдите количество информации на 10 страницах текста (на каждой странице 32 строки по 64 символа) при использовании алфавита из 256 символов.

Решение. Сначала определим информационную ёмкость одного символа. Так как $256 = 2^8$, один символ несёт $i = 8$ бит, или 1 байт информации.

Вычислим общее количество символов L. На одной странице содержится

$$32 \cdot 64 = 2^5 \cdot 2^6 = 2^{11} \text{ символов,}$$

тогда во всей книге — $L = 10 \cdot 2^{11}$ символов. Информационный объём текста равен

$$I = L \cdot i = 10 \cdot 2^{11} \cdot 1 \text{ байт} = 10 \cdot 2^{11} \cdot (1/2^{10} \text{ Кбайт}) = 20 \text{ Кбайт}.$$

Ответ: 20 Кбайт.

Задача 2. В некоторой стране автомобильный номер длиной 6 символов составляется из прописных букв (всего используется 12 букв) и десятичных цифр в любом порядке. Каждый символ кодируется одинаковым и минимально возможным количеством бит, а каждый номер — одинаковым и минимально возможным целым количеством байт. Определите объём памяти в байтах, необходимый для хранения 10 автомобильных номеров.

Решение. Для записи номера используется 12 различных букв и 10 цифр (0..9), так что алфавит состоит из 22 знаков. Для кодирования одного знака требуется не менее 5 бит ($2^4 = 16 < 22 \le 32 = 2^5$), поэтому на весь номер необходимо $6 \cdot 5 = 30$ бит.

По условию на каждый номер выделяется целое число байт, поэтому берём ближайшее значение, которое содержит не менее 30 бит, это 4 байта (32 бита). Для хранения 10 номеров нужно выделить $10 \cdot 4 = 40$ байт.

Ответ: 40 байт.

Задача 3. Для регистрации на сайте некоторой страны пользователю требуется придумать пароль длиной не более 11 символов. В качестве символов используются десятичные цифры и 12 различных букв местного алфавита, причём все буквы используются в двух регистрах: как строчные, так и прописные (регистр буквы имеет значение!). Для хранения каждого такого пароля на компьютере отводится минимально возможное и одинаковое целое количество байт, при этом используется посимвольное кодирование, и все символы кодируются одинаковым и минимально возможным количеством бит. Определите объём памяти в байтах, который занимают 60 паролей.

Решение. По условию в пароле можно использовать 10 цифр (0..9) + 12 прописных букв местного алфавита + 12 строчных букв, всего $10 + 12 + 12 = 34$ символа. Для кодирования одного из 34 символов нужно выделить 6 бит памяти, так как $2^5 = 32 < 34 \le 2^6 = 64$ (5 бит не хватает, а 6 — достаточно).

Для хранения всех 11 символов пароля нужно $11 \cdot 6 = 66$ бит. Поскольку пароль должен занимать целое число байт, берём ближайшее значение, содержащее не меньшее, чем 66 бит: это 9 байт (72 бита). На 60 паролей нужен выделить $60 \cdot 9 = 540$ байт.

Ответ: 540 байт.

Выводы

- Алфавитный подход к оценке количества информации состоит в следующем:
  1) определяем мощность алфавита M;
  2) по таблице степеней числа 2 определяем минимальное количество бит информации i, приходящихся на каждый символ сообщения, так чтобы выполнилось условие $2^i \geq M$;
  3) умножаем i на число символов в сообщении L, это и есть полное количество информации: $I = L \cdot i$.
- При использовании алфавитного подхода считается, что каждый символ несёт одинаковое количество информации. Частоты встречаемости символов и сочетаний символов не учитываются.
- Количество информации определяется только длиной сообщения и мощностью алфавита.

Нарисуйте в тетради интеллект-карту этого параграфа.

Вопросы и задания

1. Приведите примеры ситуаций, когда смысл информации особого значения не имеет, а важен только её объём.
2. Какие утверждения справедливы для алфавитного подхода:
   а) количество информации зависит от длины сообщения;
   б) количество информации зависит от мощности алфавита;
   в) чем больше мощность алфавита, тем больше количество информации;
   г) важен смысл сообщения;
   д) сообщение должно быть понятно для приёмника;
   е) разные символы могут нести разное количество информации.
3. Технический документ перевели с одного языка на другой (считаем, что это было сделано максимально близко к тексту). Изменился ли смысл документа? Изменился ли его объём?
4. Как вы думаете, почему компьютеру легко извлечь несколько предложений с конкретных страниц документа, но трудно составить аннотацию к нему?
5. *Работа в парах*. Придумайте задачу на вычисление количества информации и решите её. Затем предложите напарнику сделать то же самое. Сравните ваши результаты.

§ 8
Системы счисления

Ключевые слова:

- система счисления
- позиционная система
- основание
- разряд
- развёрнутая форма записи числа
- схема Горнера

Система счисления — это правила записи чисел с помощью специальных знаков — цифр, а также соответствующие правила выполнения операций с этими числами.
Позиционная система счисления — это такая система, в которой значение цифры (её вес) полностью определяется её местом (позицией) в записи числа.
Алфавит системы счисления — это используемый в ней набор цифр.
Основание — это количество цифр в алфавите (мощность алфавита).
Разряд — это позиция цифры в записи числа. Разряды в записи целых чисел нумеруются с нуля справа налево.

Целые числа

В быту мы используем краткую форму записи чисел в десятичной системе счисления, например 6375. В такой записи 6 стоит в третьем разряде и обозначает тысячи (10^3), 3 — во втором разряде (сотни, 10^2), 7 — в первом (десятки, 10^1), а 5 — в нулевом (единицы, 10^0). Не забывайте, что любое число (кроме нуля!) в нулевой степени равно 1. Поэтому

$$6375 = 6 \cdot 10^3 + 3 \cdot 10^2 + 7 \cdot 10^1 + 5 \cdot 10^0.$$

Это **развёрнутая форма** записи числа.

Это же число можно представить в другой форме, которая называется **схемой Горнера**:

$$6375 = ((6 \cdot 10 + 3) \cdot 10 + 7) \cdot 10 + 5.$$

С помощью схемы Горнера можно найти значение числа, не используя возведение в степень, только с помощью операций умножения и сложения.

Аналогичные выражения можно получить и в общем виде, для чисел, записанных в любой системе счисления. Пусть основание системы счисления равно $p > 1$. Её алфавит состоит из p

цифр[1] от 0 до $p-1$, т. е. «старшая» (наибольшая) цифра в позиционной системе счисления на единицу меньше, чем основание.

Рассмотрим четырёхзначное число $a_3a_2a_1a_0$, записанное в системе счисления с основанием p. Здесь a_3, a_2, a_1 и a_0 — это отдельные цифры, стоящие соответственно в третьем, втором, первом и нулевом разрядах. Это число может быть записано в развёрнутой форме

разряды → 3 2 1 0

$$a_3a_2a_1a_0 = a_3 \cdot p^3 + a_2 \cdot p^2 + a_1 \cdot p^1 + a_0 \cdot p^0$$

или с помощью схемы Горнера:

$$a_3a_2a_1a_0 = ((a_3p + a_2) \cdot p + a_1) \cdot p + a_0.$$

Из этих формул следует, что a_0 — это остаток от деления исходного числа на основание p. Поэтому если запись числа в системе с основание p закачивается на 0, то это число нацело делится на p, если заканчивается на два нуля ($a_1 = a_0 = 0$) — делится на p^2, и т. д.

Развёрнутую форму записи числа и схему Горнера можно использовать для **перевода числа в десятичную систему**. Например, пусть число 12345 записано в пятеричной системе счисления (системе с основанием 5). Нижний индекс 5 в записи 1234_5 обозначает основание системы счисления (для десятичной системы основание не указывают). Тогда

$$1234_5 = 1 \cdot 5^3 + 2 \cdot 5^2 + 3 \cdot 5^1 + 4 \cdot 5^0 = \\ = 125 + 2 \cdot 25 + 3 \cdot 5 + 4 = 194$$

$$1234_5 = ((1 \cdot 5 + 2) \cdot 5 + 3) \cdot 5 + 4 = \\ = (7 \cdot 5 + 3) \cdot 5 + 4 = 38 \cdot 5 + 4 = 194$$

Развёрнутую запись числа можно использовать для обратного перехода **от десятичной системы к системе с основанием p**. Вспомним, что a_0 — это остаток от деления исходного числа на основание p. Если мы разделим исходное число на p и отбросим остаток, мы «отбросим» последнюю цифру числа и получим

$$a_3a_2a_1 = a_3 \cdot p^2 + a_2 \cdot p + a_1.$$

Теперь легко найти a_1 — это последняя цифра получившегося числа, которая, как мы знаем, равна остатку от его деления на p. Снова разделив на p и отбросив остаток, получим число

$$a_3a_2 = a_3 \cdot p + a_2,$$

из которого найдём a_2 как остаток от деления на p. Разделив на p ещё раз, получаем последнюю цифру a_3.

---

[1] При $p > 10$ используются также и латинские буквы.

Переведём, например, число 194 в пятеричную систему счисления ($p = 5$). Найдём остаток от деления на 5:

$$194 = 38 \cdot 5 + 4.$$

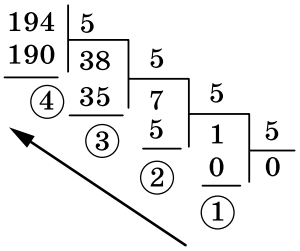
Таким образом, мы нашли последнюю цифру — 4. Частное равно 38, повторяем ту же операцию:

$$38 = 7 \cdot 5 + 3.$$

Следующая (с конца) цифра числа — 3. Дальше получаем

$$7 = 1 \cdot 5 + 2,$$

третья с конца цифра — 2, а четвёртая — 1 (единица уже не делится на 5). Обратим внимание, что с помощью этого способа мы находим цифры числа, начиная с последней. Поэтому полученные остатки нужно выписать в обратном порядке:



Ответ: 1234_5.

Для перевода числа из десятичной системы в систему счисления с основанием p нужно делить это число на p, отбрасывая остаток на каждом шаге, пока не получится 0. Затем остаётся выписать найденные остатки в обратном порядке.

Такой алгоритм фактически использует схему Горнера, «раскручивая» её в обратном порядке. При каждом делении частное и остаток определяются однозначно, поэтому представление числа в любой позиционной системе единственно.

Рассмотренные приёмы позволяют записать любое неотрицательное число в заданной позиционной системе счисления. Признаком отрицательного числа служит знак «—», после которого по тем же правилам записывается модуль числа.

Задача 1. Число 71 в системе счисления с основанием x записывается как 56_x. Найдите x.

Решение. Представим это число в развёрнутой форме:

$$71 = 56_x = 5 \cdot x^1 + 6 \cdot x^0 = 5 \cdot x + 6.$$

Решая уравнение $71 = 5x + 6$ относительно неизвестного x, получаем $x = 13$. Значит, искомое основание системы — 13.

Ответ: 13.

Задача 2. Решите уравнение $16_x + 33_x = 52_x$.

Решение. Представим все числа в развёрнутой форме:

$$16_x = x + 6; \quad 33_x = 3 \cdot x + 3; \quad 52_x = 5 \cdot x + 2.$$

Тогда исходное уравнение приводится к виду $4 \cdot x + 9 = 5 \cdot x + 2$, его решение: $x = 7$.

Ответ: 7.

Задача 3. Число 71 в системе счисления с основанием x записывается как 155_x. Найдите x.

Решение. Представим это число в развёрнутой форме:

$$71 = 155_x = 1 \cdot x^2 + 5 \cdot x^1 + 5 \cdot x^0 = x^2 + 5 \cdot x + 5.$$

Квадратное уравнение $71 = x^2 + 5x + 5$ имеет два решения: $x_1 = -11$ и $x_2 = 6$. Искомое основание системы счисления положительно, поэтому правильный ответ — 6.

Ответ: 6.

Задача 4. Найдите все основания систем счисления, в которых запись числа 24 оканчивается на 3.

Решение. Здесь удобно использовать схему Горнера, из которой следует

$$24 = k \cdot x + 3,$$

где x — неизвестное основание системы счисления, а k — некоторое натуральное число или 0. Отсюда сразу получаем $21 = k \cdot x$, т. е. все интересующие нас основания являются делителями числа 21. Это могут быть основания 3, 7 и 21. Поскольку последняя цифра числа — 3, основание не может быть равно 3 (в троичной системе нет цифры 3), поэтому условию задачи удовлетворяют только основания 7 и 21.

Ответ: 7 и 21.

Задача 5. Найдите все десятичные числа, не превосходящие 40, запись которых в системе счисления с основанием 4 оканчивается на 11.

Решение. Используя схему Горнера, находим, что все интересующие нас числа имеют вид

$$N = k \cdot 4^2 + 1 \cdot 4 + 1 = k \cdot 16 + 5,$$

где k — некоторое натуральное число или 0. Подставляя $k = 0, 1, 2, 3, \ldots$, находим соответствующие числа $N = 5, 21, 37, 53, \ldots$. Из них только 5, 21 и 37 удовлетворяют условию (не больше 40).

Ответ: 5, 21 и 37.

Задача 6. Все 5-буквенные слова, составленные из букв «А», «О», «У», записаны в алфавитном порядке. Вот начало списка:

1. ААААА
2. ААААО
3. ААААУ
4. АААОА

…

Найдите слово, которое стоит на 140-м месте от начала списка.

Решение. Как ни странно, эта задача прямо связана с позиционными системами счисления. В словах используется набор из трёх разных символов, для которых задан порядок (алфавитный). Заменив буквы «А», «О» и «У» соответственно на цифры 0, 1 и 2 выпишем начало списка:

1. 00000
2. 00001
3. 00002
4. 00010

Это числа, записанные в троичной системе счисления в порядке возрастания. Тогда легко понять, что на 140-м месте от начала списка стоит число 139, записанное в троичной системе счисления:

$$139 = 12011_3.$$

Заменив обратно цифры на буквы, получаем ответ: 12011 → ОУАОО.

Ответ: ОУАОО.

Задача 7. Значение арифметического выражения $9^{2017} + 3^{2015} - 9$ записали в системе счисления с основанием 3. Определите, сколько цифр 0, 1 и 2 содержится в этой записи.

Решение. Сначала вспомним, что в десятичной системе число 10^N записывается как единица и N нулей: $1\underbrace{00\ldots0}_{N}$, число $10^N - 1$

записывается как N девяток: $\underbrace{99\ldots9}_{N}$, а число $10^N - 10^M$ — как $N - M$ девяток, за которыми стоят M нулей:

$$10^N - 10^M = \underbrace{99\ldots9}_{N-M}\underbrace{00\ldots0}_{M}.$$

В другой позиционной системе получим подобные результаты, только вместо девяток нужно использовать старшую цифру системы счисления (в троичной системе — цифру 2):

$$3^N = \underbrace{10\ldots0}_{N}{}_3, \quad 3^N - 1 = \underbrace{2\ldots2}_{N}{}_3, \quad 3^N - 3^M = \underbrace{2\ldots2}_{N-M}\underbrace{0\ldots0}_{M}{}_3.$$

Теперь вернёмся к нашей задаче. Запишем выражение через степени числа 3:

$$9^{2017} + 3^{2015} - 9 = (3^2)^{2017} + 3^{2015} - 3^2 = 3^{4034} + 3^{2015} - 3^2.$$

Слагаемое 3^{4034} даст в троичной записи одну единицу, а разность $3^{2015} - 3^2$ представляет собой $2015 - 2 = 2013$ цифр 2.

Общая длина троичной записи числа $9^{2017} + 3^{2015} - 9$ определяется старшим слагаемым 3^{4034}, она равна 4035 (число 3^{4034} записывается в троичной системе как единица, за которой следуют 4034 нуля). Поэтому число нулей в этой записи равно $4035 - 1 - 2013 = 2021$.

Ответ: 2021 ноль, одна единица и 2013 двоек.

Дробные числа

Дробные числа сначала рассмотрим на примере десятичной системы. Число 0,6375 можно представить в виде

$$0{,}6375 = 6 \cdot 0{,}1 + 3 \cdot 0{,}01 + 7 \cdot 0{,}001 + 5 \cdot 0{,}0001.$$

Все множители, на которые умножаются значения цифр, представляют собой *отрицательные* степени числа 10 — основания системы счисления. То есть можно использовать развёрнутую форму записи, вводя отрицательные разряды:

разряды → −1 −2 −3 −4

$$0{,}6\ \ 3\ \ 7\ \ 5 = 6 \cdot 10^{-1} + 3 \cdot 10^{-2} + 7 \cdot 10^{-3} + 5 \cdot 10^{-4}.$$

Это число можно представить также с помощью схемы Горнера:

$$0{,}6375 = 10^{-1} \cdot (6 + 10^{-1} \cdot (3 + 10^{-1} \cdot (7 + 10^{-1} \cdot 5))).$$

Рассмотрим дробное число $0{,}a_1a_2a_3a_4$, записанное в системе счисления с основанием p. Здесь a_1, a_2, a_3, a_4 — это отдельные

цифры, стоящие соответственно в разрядах –1, –2, –3 и –4. Это число может быть записано в развёрнутой форме

$$\begin{matrix}\text{разряды} \rightarrow & -1 & -2 & -3 & -4 \\ 0,a_1 & a_2 & a_3 & a_4 & \end{matrix} = a_1 \cdot p^{-1} + a_2 \cdot p^{-2} + a_3 \cdot p^{-3} + a_4 \cdot p^{-4}$$

или с помощью схемы Горнера:

$$0,a_1a_2a_3a_4 = p^{-1} \cdot (a_1 + p^{-1} \cdot (a_2 + p^{-1} \cdot (a_3 + p^{-1} \cdot a_4))).$$

Умножив это число на p, получаем $a_1,a_2a_3a_4$. Если взять целую часть результата, мы получим цифру a_1. Таким же способом можно найти оставшиеся цифры дробной части: на каждом шаге берём дробную часть, умножаем её на p и запоминаем *целую часть* результата — это и будет очередная цифра записи числа в системе с основанием p. Например, переведём число 0,9376 в пятеричную систему:

Вычисления	Целая часть	Дробная часть
$0,9376 \cdot 5 = 4,688$	4	0,688
$0,688 \cdot 5 = 3,44$	3	0,44
$0,44 \cdot 5 = 2,2$	2	0,2
$0,2 \cdot 5 = 1$	1	0

Чтобы получить ответ, нужно выписать все целые части результатов, полученные на каждом шаге:

$$0,9376 = 0,4321_5.$$

Вычисления заканчиваются, когда при очередном умножении дробная часть результата становится равной нулю. Это означает, что все остальные цифры дробной части — нули. Всегда ли это произойдёт? К сожалению, нет. Чтобы убедиться в этом, вы можете перевести в пятеричную систему число 0,3 (должна получиться бесконечная дробь). Такая ситуация может случиться в любой системе счисления (например, вспомните, что число $^1/_3$ записывается в виде бесконечной десятичной дроби).

Если нужно перевести в другую систему число, в котором есть целая и дробная части, эти части переводят отдельно, а потом соединяют. Например, переведём число 25,375 в шестеричную систему:

$$25,375 = 25 + 0,375,$$

$$25 = 41_6, \quad 0,375 = 0,213_6 \quad \Rightarrow \quad 25,375 = 41,213_6.$$

Выводы

- Для перевода числа из системы счисления с основанием p в десятичную систему можно использовать запись числа в развёрнутой форме или схему Горнера.
- Для перевода целого числа из десятичной системы в систему счисления с основанием p нужно делить это число на p, отбрасывая остаток на каждом шаге, пока не получится 0, и затем выписать найденные остатки в обратном порядке.
- Для перевода дробной части числа из десятичной системы в систему счисления с основанием p нужно умножать её на p, отбрасывая целую часть на каждом шаге, пока не получится 0 или не будет достигнута требуемая точность. Затем нужно выписать отброшенные целые части в том порядке, в котором они были получены.

Вопросы и задания

1. Как связан в позиционной системе вес цифры и разряд, в котором она стоит?
2. Чем хороша схема Горнера с точки зрения вычислений?
3. Как перевести число из любой позиционной системы в десятичную?
4. Какие цифры входят в алфавит девятеричной системы?
5. Как вы думаете, можно ли использовать систему счисления с основанием 1000000? В чём могут быть проблемы?
6. Сформулируйте алгоритм перевода числа из семеричной системы в десятичную.
7. Сформулируйте алгоритм перевода числа из десятичной системы в семеричную.
8. Как по записи числа в пятеричной системе сразу увидеть, делится ли оно на 5? На 25? На 125?
9. Для каждого из чисел запишите следующее число (большее данного на единицу) в той же системе счисления:
   10111_2, 222_3, 323_4, 1234_5, 35555_6, 456_7, 77_8.
10. Сформулируйте алгоритм перевода дробной части десятичного числа в шестеричную систему счисления.
11. Как вы думаете, почему не все конечные десятичные дроби можно представить в виде конечных дробей в других системах счисления?
12. Какие дробные десятичные числа можно записать в виде конечной дроби в шестеричной системе счисления? Ответ обоснуйте.

Проекты

а) Схема Горнера в вычислениях
б) Программа для перевода смешанных чисел в другую систему счисления

§ 9
Двоичная система счисления

Ключевые слова:

- метод подбора
- кодирование дробных чисел

Алфавит **двоичной системы счисления** состоит из двух цифр: 0 и 1.
Все данные в компьютерных устройствах хранятся и обрабатываются как числа, представленные в двоичной системе счисления.
Для построения двоичной записи числа можно использовать общий способ (деление на 2 и выписывание остатков от деления в обратном порядке).

Перевод чисел в двоичную систему

Для перевода числа в двоичную систему счисления удобно использовать **метод подбора**, или **табличный метод** (разложение на сумму степеней двойки). Так, в числе 77 старшая степень двойки — это $64 = 2^6$ (следующая степень, $128 = 2^7$, уже больше, чем 77), поэтому

$$77 = 2^6 + 13.$$

Теперь выделяем старшую степень двойки в числе 13: это $8 = 2^3$, так что

$$77 = 2^6 + 2^3 + 5.$$

Выделяем старшую степень двойки в числе 5: это $4 = 2^2$, получаем

$$77 = 2^6 + 2^3 + 2^2 + 1 = 2^6 + 2^3 + 2^2 + 2^0.$$

Мы разложили число на сумму степеней двойки. Для «полного комплекта» здесь не хватает 2^5, 2^4 и 2^2, но можно считать, что эти степени умножаются на ноль:

$$77 = \textcircled{1}\cdot 2^6 + \textcircled{0}\cdot 2^5 + \textcircled{0}\cdot 2^4 + \textcircled{1}\cdot 2^3 + \textcircled{1}\cdot 2^2 + \textcircled{0}\cdot 2^1 + \textcircled{1}\cdot 2^0.$$

Это — развёрнутая запись числа в двоичной системе счисления, поэтому краткая запись состоит из цифр, обведённых кружками. Единицы стоят в шестом, третьем, втором и нулевом разрядах:

$$\begin{matrix} & & & & & & 6\,5\,4\,3\,2\,1\,0 \leftarrow \text{разряды} \\ 77 = & 1 & 0 & 0 & 1 & 1 & 0 & 1_2. \end{matrix}$$

Обратите внимание, что число, равное 2^N, в двоичной системе записывается как единица, за которой следуют N нулей.

Для перевода из двоичной системы в десятичную можно использовать сложение степеней двойки, соответствующих единичным разрядам:

$$\begin{matrix} \text{разряды} \rightarrow & & & & & & 6\,5\,4\,3\,2\,1\,0 \\ 1001101_2 = & 2^6 & + & 2^3 & + & 2^2 & + & 2^0 = 64 + 8 + 4 + 1 = 77. \end{matrix}$$

Кроме того, иногда удобно применять схему Горнера. В первом столбце таблицы записывают цифры двоичного числа, начиная со старшей. Вычисления начинаются с 1 (старший разряд всегда равен 1, если число — не ноль). В каждой из следующих строк результат, полученный в предыдущей строке, умножается на 2 и к нему добавляется значение очередной цифры двоичного числа (из первой ячейки той же строки):

Цифра числа	Вычисления	Результат
1	1	
0	$1 \cdot 2 + 0$	2
0	$2 \cdot 2 + 0$	4
1	$4 \cdot 2 + 1$	9
1	$9 \cdot 2 + 1$	19
0	$19 \cdot 2 + 0$	38
1	$38 \cdot 2 + 1$	77

Арифметические операции

Двоичные числа, как и десятичные, можно складывать в столбик, начиная с младшего разряда. При этом используют следующие правила (таблицу сложения):

$$0 + 0 = 0, \quad 1 + 0 = 1, \quad 1 + 1 = 10_2, \quad 1 + 1 + 1 = 11_2.$$

В двух последних случаях, когда сумма $2 = 10_2$ или $3 = 11_2$ не может быть записана с помощью одного разряда, происходит перенос в следующий разряд.

Например, сложим в столбик 10110_2 и 111011_2. Единицы сверху обозначают перенос из предыдущего разряда:

$$\begin{array}{r} {\scriptstyle 1\,1\,1\,1\,1} \\ 10110_2 \\ +\ 111011_2 \\ \hline 1010001_2 \end{array}$$

Вычитание выполняется почти так же, как и в десятичной системе. Вот основные правила:

$$0 - 0 = 0,\ 1 - 0 = 1,\ 1 - 1 = 0,\ 10_2 - 1 = 1.$$

В последнем случае приходится брать заём из предыдущего разряда. Именно этот вариант представляет наибольшие сложности, поэтому мы рассмотрим его подробно.

Чтобы понять принцип, временно вернёмся к десятичной системе. Вычтем в столбик из числа 21 число 9:

$$\begin{array}{r} 21 \\ -\ 9 \\ \hline ? \end{array}$$

Поскольку из 1 нельзя вычесть 9, нужно взять заём из предыдущего разряда, в котором стоит 2. В результате к младшему разряду добавляется 10 (основание системы счисления), а в следующем 2 уменьшается до 1. Теперь можно выполнить вычитание: $1 + 10 - 9 = 2$. В старшем разряде вычитаем из оставшейся единицы ноль:

$$\begin{array}{r} {\scriptstyle \bullet\quad\ \ } \\ {\scriptstyle 1\,10} \\ -\ 2\!\!\not{1} \\ 9 \\ \hline 12 \end{array}$$

Здесь точкой сверху обозначен разряд, из которого берётся заём.

Более сложный случай — заём из дальнего (не ближайшего) разряда. Вычтем 9 из 2001. В этом случае занять из ближайшего разряда не удаётся (там 0), поэтому берём заём из того разряда, где стоит цифра 2. Все промежуточные разряды в результате заполняются цифрой 9, это старшая цифра десятичной системы счисления:

$$\begin{array}{r} {\scriptstyle \bullet\quad\ \ \ \ } \\ {\scriptstyle 1\,9\,9\,10} \\ -\ 2\!\!\not{0}\!\!\not{0}\!\!\not{1} \\ 9 \\ \hline 1992 \end{array}$$

Что изменится в двоичной системе? Когда берётся заём, в «рабочий» разряд добавляется уже не 10, а $10_2 = 2$ (основание

системы счисления), а все «промежуточные» разряды (между «рабочим» и тем, откуда берется заём) заполняются не девятками, а единицами (старшей цифрой системы счисления). Например:

$$\begin{array}{r} \overset{\bullet}{0}112 \\ 1\cancel{1}000_2 \\ -\quad 1_2 \\ \hline 10111_2 \end{array} \qquad \begin{array}{r} \overset{\bullet}{0}11\overset{\bullet}{2}02 \\ 1\cancel{0}\cancel{0}\cancel{0}101_2 \\ -\quad 11011_2 \\ \hline 101010_2 \end{array}$$

Если требуется вычесть бóльшее число из меньшего, вычитают меньшее из большего и ставят у результата знак «минус»:

$$\begin{array}{r} -\ 11011_2 \\ 110101_2 \\ \hline ? \end{array} \rightarrow \begin{array}{r} -\ 110101_2 \\ 11011_2 \\ \hline 11010_2 \end{array} \rightarrow \begin{array}{r} -\ 11011_2 \\ 110101_2 \\ \hline \boxed{-\ 11010_2} \end{array}$$

Умножение и деление столбиком в двоичной системе выполняются практически так же, как и в десятичной системе (но с использованием правил двоичного сложения и вычитания):

$$\begin{array}{r} \times\ 10101_2 \\ \quad 101_2 \\ \hline + \quad 10101_2 \\ \quad 10101_2 \\ \hline 1101001_2 \end{array} \qquad \begin{array}{r|l} -\ 10101_2 & 111_2 \\ \cline{2-2} \quad 111_2 & 11_2 \\ \cline{1-1} \quad 111_2 & \\ \cline{1-1} \quad 0 & \end{array}$$

Сложение и вычитание степеней числа 2

Задача 1. Постройте двоичную запись числа $2^{12} + 2^7 - 2^5$.

Решение. Вспомним, что число 2^N в двоичной системе записывается как единица, за которой следуют N нулей:

$$2^N = 10\underbrace{\ldots0}_{N}.$$

Построим число $2^N - 2^M$ при $M < N$, используя правила вычитания:

$$\begin{array}{r} \overbrace{}^{N} \\ 100\ldots00\overbrace{0000}^{M}{}_2 \\ -10\ldots0_2 \\ \hline \underbrace{11\ldots11}_{N-M}0\ldots0_2 \end{array}$$

Видим, что это значение записывается в двоичной системе как $N - M$ единиц, за которыми следуют M нулей. Поэтому в нашем примере $2^7 - 2^5 = 1100000_2$. Кроме того, слагаемое 2^{12} добавляет единицу в 12-м разряде, так что

$$2^{12} + 2^7 - 2^5 = 1000001100000_2.$$

Задача 2. Сколько единиц содержит двоичная запись числа

$$8^{12} + 4^{34} - 2^{18} + 120?$$

Решение. Сначала представим все числа как степени числа 2. Учитывая, что

$$120 = 128 - 8 = 2^7 - 2^3,$$

получаем

$$(2^3)^{12} + (2^2)^{34} - 2^{18} + 2^7 - 2^3 = 2^{68} + 2^{36} - 2^{18} + 2^7 - 2^3.$$

Как показано выше, пара $2^{36} - 2^{18}$ даёт $36 - 18 = 18$ единиц, пара $2^7 - 2^3$ добавляет четыре единицы, и ещё одну единицу добавляет слагаемое 2^{68}. Поэтому заданное число содержит $18 + 4 + 1 = 23$ единицы.

Ответ: 23 единицы.

Задача 3. Сколько значащих нулей содержит двоичная запись числа

$$8^{128} + 4^{344} - 2^{136} - 70?$$

Решение. Для того чтобы определить число значащих нулей, можно вычесть количество единиц из общей длины двоичной записи числа. Запишем все слагаемые как степени двойки, представив число 70 в виде $70 = 64 + 8 - 2 = 2^6 + 2^3 - 2^1$:

$$(2^3)^{128} + (2^2)^{344} - 2^{136} - 2^6 - 2^3 + 2^1 =$$
$$= 2^{688} + 2^{384} - 2^{136} - 2^6 - 2^3 + 2^1.$$

Старшая степень двойки равна 688, к ней добавляется некоторое число, поэтому запись числа в двоичной системе содержит 689 цифр (единиц и нулей).

Теперь рассмотрим «цепочку вычитания» $2^{384} - 2^{136} - 2^6 - 2^3$. Разность $2^{384} - 2^3$ в двоичной системе счисления запишется как $384 - 3 = 381$ единица, за которой следуют 3 нуля, при вычитании 2^{136} и 2^6 две из 381 единиц заменяются на нули, так что остаётся 379 единиц.

Слагаемые $2^{688} + 2^1$ дают ещё две единицы, так что общее число единиц равно 381, а число значащих нулей равно $689 - 381 = 308$.

Ответ: 308 значащих нулей.

Дробные числа

Для перевода дробного числа в двоичную систему используется общий подход: нужно умножать число на 2, запоминать целую часть и отбрасывать её перед следующим умножением. Например, для числа 0,8125 получаем:

Вычисления	Целая часть	Дробная часть
$0{,}8125 \cdot 2 = 1{,}625$	1	0,625
$0{,}625 \cdot 2 = 1{,}25$	1	0,25
$0{,}25 \cdot 2 = 0{,}5$	0	0,5
$0{,}5 \cdot 2 = 1$	1	0

Таким образом, $0{,}8125 = 0{,}1101_2$.

Давайте посмотрим, как хранится в памяти число 0,6. Выполняя умножение на 2 и выделение целой части, мы получим периодическую бесконечную дробь:

$$0{,}6 = 0{,}100110011001_2\ldots = 0{,}(1001)_2.$$

Это значит, что для записи десятичного числа 0,6 в двоичной системе счисления требуется бесконечное число разрядов. Поскольку реальный компьютер не может иметь бесконечную память, число 0,6 в двоичном представлении хранится в памяти с ошибкой[1)] (погрешностью).

Большинство дробных чисел хранится в памяти с некоторой ошибкой. При выполнении вычислений с дробными числами ошибки накапливаются и могут существенно влиять на результат.

Отметим, что эта проблема связана не с двоичной системой, а с ограниченным размером ячейки памяти компьютера, отведённой на хранение числа. В любой системе счисления существуют бесконечные дроби, которые не могут быть точно представлены конечным числом разрядов.

1) Последний стандарт кодирования вещественных чисел IEEE 754-2008 позволяет записывать в память числа в десятичном виде. Однако вычисления с такими данными выполняются сложнее и медленнее, чем с двоичными. Кроме того, проблема конечного числа разрядов остаётся: десятичная дробь, равная $^1/_3$, по-прежнему не может быть точно представлена в памяти компьютера.

Обеспечение точности расчётов с дробными (вещественными) числами — это очень важная и актуальная проблема, пока до конца не решённая. Поэтому сначала надо попытаться решить задачу, используя только операции с целыми числами. Например, пусть требуется проверить, верно ли, что $A < \sqrt{B}$, где A и B — целые числа. При извлечении квадратного корня мы сразу переходим в область вещественных чисел, где могут возникнуть вычислительные ошибки. Вместо этого можно возвести обе части неравенства в квадрат и проверять равносильное условие $A^2 < B$, используя только операции с целыми числами.

Если же всё-таки нужно обязательно использовать дробные числа и нельзя жертвовать точностью, приходится хранить их в нестандартном виде, например в виде отношения целых чисел (например, $0{,}6 = 6/10$) и вычислять отдельно числители и знаменатели простых дробей, переходя к вещественным числам только при выводе конечного результата. Этот подход применяется в программных системах символьных вычислений, например, в *Maple* **(www.maplesoft.com)** и *Mathematica* **(www.wolfram.com)**. Однако выполнение таких расчётов занимает очень много времени.

Двоичная система: достоинства и недостатки

Двоичная система служит основой всех расчётов в современных компьютерах. Она обладает следующими **преимуществами**:

- для того чтобы построить компьютер, работающий с двоичными данными, достаточно иметь **устройства с двумя состояниями** (включено/выключено); первыми такими устройствами были электромагнитные реле, сейчас применяются микроэлектронные элементы;
- **надёжность и защита от помех** при передаче информации (для приёма двоичного кода не нужно измерять сигнал, а достаточно знать, какое из двух значений он принимает в каждый заданный момент времени);
- компьютеру **проще выполнять вычисления** с двоичными числами, нежели с десятичными; например, умножение фактически сводится к многократному сложению, а деление — к вычитанию.

Тем не менее, с точки зрения человека, у двоичной системы есть **недостатки**:

- двоичная запись чисел получается **длинной**: например, число 1024 записывается в виде 10000000000_2 — здесь легко перепутать количество идущих подряд нулей;
- запись **однородна**, т. е. содержит только нули и единицы; поэтому при работе с двоичными числами легко ошибиться или запутаться.

Выводы

- Чтобы перевести число в двоичную систему, нужно представить его как сумму степеней числа 2.
- Чтобы перевести число из двоичной системы в десятичную, нужно записать его в развёрнутой форме как сумму степеней числа 2 и выполнить сложение.
- Последняя цифра двоичной записи целого числа — это остаток от деления этого числа на 2. Двоичная запись чётного числа заканчивается на 0, а нечётного — на 1. Если число делится на 4, то его двоичная запись заканчивается на два нуля.
- Большинство дробных чисел хранится в памяти с некоторой ошибкой. При выполнении вычислений с дробными числами ошибки накапливаются и могут существенно влиять на результат.
- Человеку неудобно работать с данными, записанными в двоичной системе счисления, потому что запись больших чисел получается длинной и однородной (содержит только 0 и 1).

Нарисуйте в тетради интеллект-карту этого параграфа.

Вопросы и задания

1. Как вы думаете, какие дробные числа могут быть точно представлены в памяти компьютера в двоичном коде?
2. Почему всегда рекомендуется выполнять вычисления, используя только операции с целыми числами (если есть такая возможность)?
3. Как можно работать с дробными числами, не теряя точности? В чём недостатки такого подхода?
4. Сравните преимущества и недостатки использования двоичной системы счисления с точки зрения человека и с точки зрения компьютера.

§ 10
Восьмеричная система счисления

Ключевые слова:

- триада

Восьмеричная система (система с основанием 8, использующая цифры от 0 до 7) применяется для краткой записи двоичных кодов.

Для перевода десятичного числа в восьмеричную используют общий алгоритм для позиционных систем (деление на 8, выписывание остатков в обратном порядке).

Для перевода числа из восьмеричной системы в десятичную значение каждой цифры умножают на 8 в степени, равной разряду этой цифры, и полученные произведения складывают:

разряды → 2 1 0

$$144_8 = 1 \cdot 8^2 + 4 \cdot 8^1 + 4 \cdot 8^0 = 64 + 4 \cdot 8 + 4 = 100.$$

Связь с двоичной системой счисления

Пусть требуется перевести число из восьмеричной системы в двоичную. Конечно, можно перевести число сначала в десятичную систему, а потом — в двоичную. Но для этого требуется выполнить две непростые операции, в каждой из них легко ошибиться.

Оказывается, можно сделать перевод из восьмеричной системы в двоичную напрямую, используя тесную связь между этими системами: их основания связаны равенством $2^3 = 8$. Покажем это на примере трёхзначного восьмеричного числа abc_8, где a, b и c — восьмеричные цифры. Запишем число abc_8 в развёрнутой форме:

$$abc_8 = a \cdot 8^2 + b \cdot 8^1 + c \cdot 8^0 = a \cdot 2^6 + b \cdot 2^3 + c \cdot 2^0.$$

Теперь переведём отдельно каждую восьмеричную цифру в двоичную систему. Поскольку три двоичных разряда позволяют записать все числа от 0 до 7 (и никаких других чисел!), каждая восьмеричная цифра может быть записана в виде **триады** (группы из трёх цифр) в двоичной системе (табл. 2.1).

$$\begin{aligned} a = (a_2a_1a_0)_2 &= a_2 \cdot 2^2 + a_1 \cdot 2^1 + a_0 \cdot 2^0, \\ b = (b_2b_1b_0)_2 &= b_2 \cdot 2^2 + b_1 \cdot 2^1 + b_0 \cdot 2^0, \\ c = (c_2c_1c_0)_2 &= c_2 \cdot 2^2 + c_1 \cdot 2^1 + c_0 \cdot 2^0. \end{aligned}$$

Таблица 2.1

0	000	4	100
1	001	5	101
2	010	6	110
3	011	7	111

Здесь a_i, b_i, и c_i ($i = 0, 1, 2$) — это двоичные разряды (0 или 1). Тогда получаем:

$$abc_8 = (a_2 \cdot 2^2 + a_1 \cdot 2^1 + a_0 \cdot 2^0) \cdot 2^6 + (b_2 \cdot 2^2 + b_1 \cdot 2^1 + b_0 \cdot 2^0) \cdot 2^3 + (c_2 \cdot 2^2 + c_1 \cdot 2^1 + c_0 \cdot 2^0) \cdot 2^0.$$

Раскрывая скобки, мы получим разложение исходного числа по степеням двойки, т. е. его запись в двоичной системе счисления:

$$abc_8 = \underbrace{a_2 \cdot 2^8 + a_1 \cdot 2^7 + a_0 \cdot 2^6}_{a} + \underbrace{b_2 \cdot 2^5 + b_1 \cdot 2^4 + b_0 \cdot 2^3}_{b} + \underbrace{c_2 \cdot 2^2 + c_1 \cdot 2^1 + c_0 \cdot 2^0}_{c}$$

Таким образом, $abc_8 = (a_2a_1a_0\,b_2b_1b_0\,c_2c_1c_0)_2$. Это значит, что каждая цифра восьмеричного числа при переводе в двоичную систему записывается как группа из трёх двоичных разрядов **(триада)** независимо от других цифр.

Алгоритм перевода восьмеричного числа в двоичную систему счисления

1. Перевести каждую цифру (отдельно) в двоичную систему. Записать результаты в виде триады, добавив, если нужно, нули в начале.
2. Соединить триады в одно «длинное» двоичное число.

Например, для числа 35721_8 сразу получаем:

$$35721_8 = 11\,101\,111\,010\,001_2.$$

В этой записи триады специально отделены друг от друга пробелами. Обратите внимание, что все триады дополнены спереди нулями до трёх цифр:

$$2 = 10_2 = 010_2, \qquad 1 = 1_2 = 001_2.$$

Для самой первой триады это делать необязательно, потому что лидирующие нули в записи числа никак его не меняют. Напротив, если «потерять» нули в середине числа, получится неверный результат.

А если разбить двоичное число на триады, то можно быстро перевести его в восьмеричную систему счисления. Именно благодаря этому свойству восьмеричную систему удобно использовать для сжатой записи двоичных кодов.

Алгоритм перевода двоичного числа в восьмеричную систему счисления

1. Разбить двоичное число на триады, *начиная справа*. В начало самой первой триады добавить лидирующие нули, если это необходимо.
2. Перевести каждую триаду (отдельно) в восьмеричную систему счисления[1)].
3. Соединить полученные цифры в одно «длинное» число.

Переведём в восьмеричную систему число 1010011100101110111_2. Разобьём его на триады (начиная справа), в начало числа нужно добавить два нуля (они подчёркнуты):

$$1010011100101110111_2 = \underline{00}1\ 010\ 011\ 100\ 101\ 110\ 111_2.$$

Далее по табл. 2.1 переводим каждую триаду отдельно в восьмеричную систему:

$$1010011100101110111_2 = 1234567_8.$$

Теперь представьте себе объём вычислений, который потребуется для решения этой задачи через десятичную систему.

Арифметические операции

При вычислениях в восьмеричной системе нужно помнить, что максимальная цифра — это 7. Перенос при сложении возникает тогда, когда сумма в очередном разряде получается больше 7.

Заём из старшего разряда равен $10_8 = 8$, а все «промежуточные» разряды заполняются цифрой 7 — старшей цифрой системы счисления. Приведём примеры сложения и вычитания:

$$\begin{array}{r} {\scriptstyle 111} \\ 356_8 \\ +\ 4662_8 \\ \hline 5240_8 \end{array} \qquad \begin{array}{l} 6+2=1\cdot 8+\textcircled{0} \\ 5+6+1=1\cdot 8+\textcircled{4} \\ 3+6+1=1\cdot 8+\textcircled{2} \\ 0+4+1=\textcircled{5} \end{array}$$

$$\begin{array}{r} {\scriptstyle \bullet\ \bullet} \\ 456_8 \\ -\ 277_8 \\ \hline 157_8 \end{array} \qquad \begin{array}{l} (6+8)-7=\textcircled{7} \\ (5-1+8)-7=\textcircled{5} \\ (4-1)-2=\textcircled{1} \end{array}$$

В примере на сложение запись $1 \cdot 8 + 2$ означает, что получилась сумма, большая 7, которая не помещается в один разряд.

[1)] Заметим, что значение цифры в восьмеричной системе счисления совпадает со значением этой же цифры в десятичной системе.

Единица идёт в перенос, а двойка остаётся в этом разряде. При выполнении вычитания запись «–1» означает, что из этого разряда раньше был заём (его значение уменьшилось на 1), а запись «+8» — заём из старшего разряда.

Применение

С помощью восьмеричной системы удобно кратко записывать содержимое областей памяти, в которых количество бит кратно трём. Например, 6-битные данные «упаковываются» в две восьмеричные цифры. Некоторые компьютеры 1960-х годов использовали 24-битные и 36-битные данные, они записывались соответственно с помощью 8 и 12 восьмеричных цифр.

Восьмеричная система использовалась для кодирования команд в компьютерах 1950–1980-х годов, например в американской серии PDP, советских компьютерах серий ДВК, СМ ЭВМ, БЭСМ.

Сейчас восьмеричная система применяется для установки прав на доступ к файлу в *Linux* (и других *Unix*-системах) с помощью команды `chmod`. Режим доступа кодируется тремя битами, которые разрешают чтение (`r`, *read*, старший бит), запись (`w`, *write*) и выполнение файла (`x`, *execute*, младший бит). Код $7 = 111_2$ (`rwx`) означает, что все биты установлены (полный доступ), а код $5 = 101_2$ (`r-x`) разрешает чтение и выполнение файла, но запрещает его изменение.

Выводы

- Восьмеричная система счисления используется для сжатой записи двоичных кодов.
- При переводе числа из восьмеричной системы в двоичную каждая цифра отдельно представляется в виде триады — группы из трёх двоичных разрядов.
- При переводе числа из двоичной системы в восьмеричную двоичная запись разбивается на триады, начиная с конца, и каждая триада отдельно переводится в одну восьмеричную цифру.

Нарисуйте в тетради интеллект-карту этого параграфа.

Вопросы и задания

1. Как вы думаете, из каких других систем счисления возможен быстрый переход к двоичной системе и обратно?
2. Сколько существует различных двузначных восьмеричных чисел? Трёхзначных восьмеричных чисел?

§ 11
Шестнадцатеричная система счисления

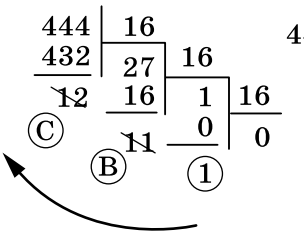
Ключевые слова:

- тетрада

Шестнадцатеричная система используется для записи адресов и содержимого ячеек памяти компьютера. Её алфавит содержит 16 цифр, вместе с 10 арабскими цифрами (0..9) используются первые буквы латинского алфавита:

$$A = 10,\ B = 11,\ C = 12,\ D = 13,\ E = 14 \text{ и } F = 15.$$

Для перевода чисел из десятичной системы в шестнадцатеричную используют деление на 16 и взятие остатков. Все остатки, бóльшие 9, нужно заменить на буквы:



$444 = 1BC_{16}$

Для обратного перевода значение каждой цифры умножают на 16 в степени, равной её разряду, и полученные значения складывают:

разряды → 2 1 0

$$1BC_{16} = 1 \cdot 16^2 + 11 \cdot 16^1 + 12 \cdot 16^0 = 256 + 176 + 12 = 444.$$

Можно также использовать схему Горнера:

$$1BC_{16} = (1 \cdot 16 + 11) \cdot 16 + 12 = 27 \cdot 16 + 12 = 444.$$

Связь с двоичной системой счисления

Основания двоичной и шестнадцатеричной систем связаны соотношением $2^4 = 16$, поэтому можно переводить числа из шестнадцатеричной системы в двоичную напрямую. Алгоритмы перевода чисел из шестнадцатеричной системы в двоичную и обратно аналогичны соответствующим алгоритмам для восьмеричной системы. Каждая шестнадцатеричная цифра представляется в виде **тетрады**

(группы из четырёх двоичных цифр) в двоичной системе счисления (табл. 2.2).

Таблица 2.2

0	0000	8	1000
1	0001	9	1001
2	0010	A (10)	1010
3	0011	B (11)	1011
4	0100	C (12)	1100
5	0101	D (13)	1101
6	0110	E (14)	1110
7	0111	F (15)	1111

Переведём в двоичную систему число $5E123_{16}$ (здесь показана разбивка на тетрады):

$$5E123_{16} = 101\,1110\,0001\,0010\,0011_2.$$

Обратите внимание, что для цифр, меньших 8 (кроме первой), результат перевода в двоичную систему нужно дополнить старшими нулями до четырёх знаков.

При обратном переводе нужно разбить двоичное число на тетрады, *начиная справа*, и каждую тетраду отдельно перевести в двоичную систему:

$$10000100001010101111100_2 =$$
$$= 10\ \ 0001\ \ 0000\ \ 1010\ \ 1011\ \ 1100_2 = \ 210ABC_{16}.$$

Перевод из шестнадцатеричной системы в восьмеричную (и обратно) удобнее выполнять через двоичную систему. Можно, конечно, использовать и десятичную систему, но в этом случае объём вычислений будет значительно больше.

Арифметические операции

При выполнении сложения нужно помнить, что в системе с основанием 16 перенос появляется тогда, когда сумма в очередном разряде превышает 15. Удобно сначала переписать исходные числа, заменив все буквы на их численные значения:

$$\begin{array}{r} A5B_{16} \\ +\ C7E_{16} \\ \hline 16D9_{16} \end{array} \qquad \begin{array}{r} {\scriptstyle 1\quad\ 1\quad\ } \\ 10\ \ 5\ \ 11 \\ +\ 12\ \ 7\ \ 14 \\ \hline 1\ 6\ 13\ \ 9 \end{array} \qquad \begin{array}{l} 11 + 14 = 1{\cdot}16 + \textcircled{9} \\ 5 + 7 + 1 = 13 = \textcircled{D} \\ 10 + 12 = 1{\cdot}16 + \textcircled{6} \\ 0 + 0 + 1 = \textcircled{1} \end{array}$$

При вычитании заём из старшего разряда равен $10_{16} = 16$, а все «промежуточные» разряды заполняются цифрой F — старшей цифрой системы счисления.

$$\begin{array}{rl} & \overset{\bullet}{12}\ \overset{\bullet}{5}\ 11 \\ \_\ \mathrm{C5B}_{16} & \_\ 12\ \ 5\ \ 11 \\ \mathrm{A7E}_{16} & 10\ \ 7\ \ 14 \\ \hline \mathrm{1DD}_{16} & 1\ \ 13\ \ 13 \end{array} \quad \begin{array}{l} (11+16)-14=13=\text{D} \\ (5-1+16)-7=13=\text{D} \\ (12-1)-10=1 \end{array}$$

Если нужно работать с числами, записанными в разных системах счисления, их сначала приводят к какой-нибудь одной системе. Например, пусть требуется сложить 53_8 и 56_{16} и записать результат в двоичной системе счисления. Здесь можно выполнять сложение в двоичной, восьмеричной, десятичной или шестнадцатеричной системах. Переход к десятичной системе, а потом перевод результата в двоичную трудоёмок. Практика показывает, что больше всего ошибок делается при вычислениях именно в двоичной системе, поэтому лучше выбирать восьмеричную или шестнадцатеричную систему. Переведём число 53_8 в шестнадцатеричную систему через двоичную:

$$53_8 = 101\ 011_2 = 10\ 1011_2 = 2\mathrm{B}_{16}.$$

Теперь выполним сложение:

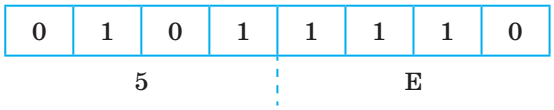
$$2\mathrm{B}_{16} + 56_{16} = 81_{16}$$

и переведём результат в двоичную систему:

$$81_{16} = 1000\ 0001_2.$$

Применение

Шестнадцатеричная система оказалась очень удобной для записи значений ячеек памяти. Байт в современных компьютерах представляет собой 8 соседних бит, т. е. две тетрады. Таким образом, значение байтовой ячейки можно записать как две шестнадцатеричные цифры:



Каждый полубайт (4 бита) «упаковывается» в одну шестнадцатеричную цифру. Благодаря этому замечательному свойству шестнадцатеричная система в сфере компьютерной техники практически полностью вытеснила восьмеричную[1)].

[1)] Начиная с 1964 года, когда шестнадцатеричная система стала широко использоваться в документации на новый компьютер IBM/360.

Выводы

- При переводе числа из шестнадцатеричной системы в двоичную каждая цифра отдельно представляется в виде тетрады — группы из четырёх двоичных разрядов.
- При переводе числа из двоичной системы в шестнадцатеричную двоичная запись разбивается на тетрады, начиная с конца, и каждая тетрада отдельно переводится в одну шестнадцатеричную цифру.
- Перевод чисел между шестнадцатеричной и восьмеричной системами удобнее выполнять через двоичную систему.

Нарисуйте в тетради интеллект-карту этого параграфа.

Вопросы и задания

1. Почему в шестнадцатеричной системе счисления появилась необходимость использовать латинские буквы?
2. Какое минимальное основание должно быть у системы счисления, чтобы в ней могли быть записаны числа 123, 4AB, 9A3 и 8455?

§ 12
Другие системы счисления

Ключевые слова:

- троичная уравновешенная система
- трит
- двоично-десятичная система

Троичная уравновешенная система счисления

В истории компьютерной техники применялись и другие системы счисления. Например, в 1958 г. была создана электронная вычислительная машина (ЭВМ) «Сетунь» (главный конструктор — Н. П. Брусенцов), которая использовала троичную систему счисления. Всего в 1960-х годах было выпущено более 50 промышленных образцов ЭВМ «Сетунь».

В троичной уравновешенной системе основание равно 3, используются три цифры: $\overline{1}$ («минус 1»), 0 и 1. Один троичный разряд называется **тритом** (в отличие от двоичного бита). Система называется **уравновешенной**, потому что с помощью любого числа разрядов можно закодировать равное число положительных и отрицательных чисел и число ноль. Вот, например, все двухразрядные числа в троичной системе (табл. 2.3).

Таблица 2.3

–4	$\bar{1}\ \bar{1}$	$= (-1) \cdot 3^1 + (-1) \cdot 3^0$
–3	$\bar{1}\ 0$	$= (-1) \cdot 3^1 + 0 \cdot 3^0$
–2	$\bar{1}\ 1$	$= (-1) \cdot 3^1 + 1 \cdot 3^0$
–1	$0\ \bar{1}$	$= 0 \cdot 3^1 + (-1) \cdot 3^0$
0	0 0	$= 0 \cdot 3^1 + 0 \cdot 3^0$
1	0 1	$= 0 \cdot 3^1 + 1 \cdot 3^0$
2	$1\ \bar{1}$	$= 1 \cdot 3^1 + (-1) \cdot 3^0$
3	1 0	$= 1 \cdot 3^1 + 0 \cdot 3^0$
4	1 1	$= 1 \cdot 3^1 + 1 \cdot 3^0$

В последнем столбце этой таблицы числа записаны в развёрнутой форме, которую можно использовать для перевода из троичной уравновешенной системы в десятичную.

Заметьте, что положительные и отрицательные числа кодируются с помощью одних и тех же правил. Это большое преимущество по сравнению с двоичным кодированием, при котором для хранения отрицательных чисел пришлось изобретать специальный код.

Троичная уравновешенная система счисления даёт ключ к решению *задачи Баше*, которая была известна ещё в XIII веке Леонардо Пизанскому (Фибоначчи):

> Найти такой набор из 4 гирь, чтобы с их помощью на чашках равноплечных весов можно было взвесить груз массой от 1 до 40 кг включительно. Гири можно располагать на любой чашке весов.

Каждая гиря может быть в трёх состояниях:

1) лежать на той же чашке весов, что и груз: в этом случае её вес вычитается из суммы ($\bar{1}$);
2) не участвовать во взвешивании (0);
3) лежать на другой чашке: её вес добавляется к сумме (1).

Поэтому веса гирь нужно выбрать равными степеням числа 3, т. е. 1, 3, 9 и 27 кг.

Двоично-десятичная система счисления

Существует ещё один простой способ записи десятичных чисел с помощью цифр 0 и 1. Этот способ называется двоично-десятичной системой (ДДС) — это нечто среднее между двоичной

и десятичной системами. На английском языке такое кодирование называется *binary coded decimal* (BCD) — десятичные числа, закодированные двоичными цифрами.

В ДДС каждая цифра десятичного числа записывается двоичными знаками. Но среди цифр 0–9 есть такие, которые занимают 1, 2, 3 и 4 двоичных разряда. Чтобы запись числа была однозначной и не надо было искать границу между цифрами, на любую цифру отводят 4 бита. Таким образом, 0 записывается как 0000, а 9 — как 1001. Например:

$$\begin{matrix} 9024{,}19 = & 1001 & 0000 & 0010 & 0100, & 0001 & 1001_{\text{ДДС}}. \\ & 9 & 0 & 2 & 4 & 1 & 9 \end{matrix}$$

При обратном переводе из ДДС в десятичную систему надо учесть, что каждая цифра занимает 4 бита, и добавить недостающие нули:

$$101010011{,}01111_{\text{ДДС}} = 0001\ 0101\ 0011,\ 0111\ 1000_{\text{ДДС}} = 153{,}78.$$

Важно помнить, что запись числа в ДДС не совпадает с его записью в двоичной системе:

$10101{,}1_{\text{ДДС}} = 15{,}8;$
$10101{,}1_2 = 16 + 4 + 1 + 0{,}5 = 21{,}5.$

Использование ДДС даёт следующие **преимущества**:

- двоично-десятичный код очень легко переводить в десятичный, например, для вывода результата на экран;
- просто выполняется умножение и деление на 10, а также округление;
- конечные десятичные дроби записываются точно, без ошибки, так что вычисления в ДДС (вместо двоичной системы) дадут тот же результат, что и ручные расчёты человека «на бумажке»; поэтому ДДС используется в калькуляторах.

Есть, однако, и **недостатки**:

- хранение чисел в ДДС требует больше памяти, чем стандартный двоичный код;
- усложняются арифметические операции.

Выводы

- В троичной уравновешенной системе основание равно 3, используются три цифры: «минус 1», 0 и 1. Один троичный разряд называется тритом.

- В двоично-десятичной системе счисления каждая цифра десятичного числа отдельно записывается в виде четырёх двоичных разрядов.

Нарисуйте в тетради интеллект-карту этого параграфа.

Вопросы и задания

1. Как поменять знак числа, записанного в уравновешенной системе?
2. Сколько положительных и отрицательных чисел можно закодировать с помощью 5 разрядов в троичной уравновешенной системе счисления?
3. Попробуйте сформулировать правила сложения чисел в троичной уравновешенной системе.
4. Сравните троичную уравновешенную систему счисления с двоичной. Как вы думаете, почему разработчики компьютеров всё-таки выбрали двоичный код для хранения данных? Найдите сведения об этом в дополнительных источниках.
5. Верно ли, что любая последовательность нулей и единиц может быть числом, закодированным в двоично-десятичной системе? Обоснуйте свой ответ.
6. Какие числа записываются одинаково в двоичной и двоично-десятичной системах счисления?
7. Сравните двоично-десятичную систему с двоичной. В чём преимущества и недостатки каждой из них?

Подготовьте сообщение

а) «Троичная уравновешенная система счисления»
б) «ЭВМ «Сетунь»»
в) «Факториальная система счисления»
г) «Фибоначчиева система счисления»

Проекты

а) Программа для работы с числами в троичной уравновешенной системе счисления
б) Программа для работы с числами в факториальной системе счисления
в) Программа для работы с числами в фибоначчиевой системе счисления

§ 13
Кодирование текстов

Ключевые слова:

- текстовый файл
- шрифт
- внедрение шрифтов
- кодировка ASCII
- кодовая страница
- UNICODE

Текст в памяти компьютера хранится как последовательность двоичных кодов символов. Изображения символов хранятся в отдельных шрифтовых файлах.

В текстовых файлах (которые не содержат оформления, например, в файлах с расширением *txt*) хранятся не изображения символов, а их коды. Откуда же компьютер берёт изображения символов, когда выводит текст на экран? Оказывается, при этом с диска загружается шрифтовой файл (он может иметь, например, расширение *fon*, *ttf*, *otf*), в котором хранятся изображения, соответствующие каждому из кодов[1)]. Именно эти изображения и выводятся на экран. Это значит, что при изменении шрифта текст, показанный на экране, может выглядеть совсем по-другому. Например, многие шрифты не содержат изображений русских букв. Поэтому, когда вы передаёте (или пересылаете) кому-то текстовый файл, нужно убедиться, что у адресата есть использованный вами шрифт. Современные текстовые процессоры умеют *внедрять* шрифты в файл; в этом случае файл содержит не только коды символов, но и шрифтовые файлы. Хотя файл увеличивается в объеме, адресат гарантированно увидит его в таком же виде, что и вы.

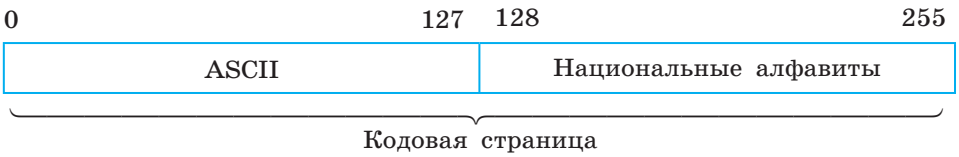
Однобайтные кодировки

Международным стандартом является 7-битная кодировка ASCII, которая определяет коды 0–127 для 128 символов, в том числе латинских букв, цифр, знаков препинания, знаков арифметических операций, скобок и др.

В современных компьютерах минимальная единица памяти, имеющая собственный адрес, — это байт (8 бит). Поэтому для хранения кодов ASCII в памяти можно добавить к ним ещё один

1) Существуют специальные программы, позволяющие создавать и редактировать шрифты, например Fontlab Studio (**http://www.fontlab.com/font-editor/fontlab-studio/**).

(старший) нулевой бит, таким образом, получая 8-битную кодировку. Дополнительный бит можно использовать: он даёт возможность добавить в таблицу ещё 128 символов с кодами от 128 до 255. Такое расширение ASCII часто называют **кодовой страницей**. Первую половину кодовой страницы (коды от 0 до 127) занимает стандартная таблица ASCII, а вторую — символы национальных алфавитов (например, русские буквы).



Для русского языка существуют несколько кодовых страниц, которые были разработаны для разных операционных систем. Наиболее известны:

- кодовая страница **Windows-1251** (CP-1251) — в системе *Windows*;
- кодовая страница **KOI8-R** — в *Unix*-совместимых операционных системах и электронной почте;
- альтернативная кодировка **CP-866** — в системе *MS DOS*;
- кодовая страница **MacCyrillic** — на компьютерах фирмы *Apple* (*Макинтош* и др.).

Проблема состоит в том, что, если набрать русский текст в одной кодировке (например, в Windows-1251), а просматривать в другой (например, в KOI8-R), текст будет очень сложно прочитать:

Windows-1251	KOI8-R
Привет, Вася!	оПХБЕР, бЮЯЪ!
рТЙЧЕФ, чБУС!	Привет, Вася!

Для веб-страниц в Интернете используют кодовые страницы Windows-1251 и KOI8-R. Браузер после загрузки страницы пытается автоматически определить её кодировку. Если ему это не удаётся, вы увидите странный набор букв вместо понятного русского текста. В этом случае нужно сменить кодировку вручную с помощью меню *Вид* браузера.

Главный недостаток однобайтных кодировок заключается в том, что в одном документе можно использовать только 256 символов (включая 128 символов ASCII).

Стандарт UNICODE

Для того чтобы в одном документе можно было использовать более 256 символов, в 1991 году был принят стандарт кодирования символов UNICODE, который включает знаки любых существующих (и даже некоторых мёртвых) языков, математические и музыкальные символы и др.

В современной версии UNICODE можно кодировать до 1 112 064 различных знаков, однако реально используются немногим более 128 000.

В системе Windows используется кодировка UNICODE, называемая UTF-16 (от англ. UNICODE Transformation Format — формат преобразования UNICODE). В ней все наиболее важные символы кодируются с помощью 16 бит (2 байт), а редко используемые — с помощью 4 байт.

В Unix-подобных системах, например в Linux, чаще применяют кодировку UTF-8. В ней все символы, входящие в таблицу ASCII, кодируются с помощью 1 байта, а другие символы могут занимать от 2 до 4 байт. Если значительную часть текста составляют латинские буквы и цифры, такой подход позволяет значительно уменьшить объём файла по сравнению с UTF-16. Текст, состоящий только из символов таблицы ASCII, кодируется точно так же, как и в кодировке ASCII. По данным поисковой системы Google, на начало 2014 года около 80% сайтов в Интернете использовали кодировку UTF-8.

Кодировки стандарта UNICODE позволяют использовать символы разных языков в одном документе, но за это приходится расплачиваться увеличением объёма файлов.

Задача 1. Автоматическое устройство осуществило перекодировку информационного сообщения на русском языке, первоначально записанного в 16-битном коде UNICODE, в 8-битную кодировку KOI8-R. При этом информационное сообщение уменьшилось на 480 бит. Какова длина сообщения в символах?

Решение. Обозначим количество символов в сообщении через L. При использовании 16-битной кодировки информационный объём сообщения равен $16 \cdot L$ бит, а после перекодирования в 8-битный код он стал равен $8 \cdot L$ бит. Таким образом, объём сообщения уменьшился на $8 \cdot L$ бит, отсюда находим: $L = 480/8 = 60$ символов.

Ответ: 60 символов.

Выводы

- В текстовых файлах содержатся только коды символов, а их изображения хранятся в памяти компьютера. Современные операционные системы загружают изображения букв из специальных шрифтовых файлов. Документы текстовых процессоров могут содержать внедрённые шрифты.
- ASCII — это 7-битная кодировка, которая является международным стандартом. Она содержит цифры, латинские буквы, скобки, знаки препинания, знаки арифметических операций и другие символы.
- Однобайтные (8-битные) кодировки (кодовые страницы) включают таблицу ASCII (символы с кодами 0..127) и дополнительную часть (символы с кодами 128...255), в которую входят символы национальных алфавитов.
- Стандарт UNICODE позволяет записывать знаки любых существующих (и даже некоторых мёртвых) языков, математические и музыкальные символы и др. (всего до 1 112 064 знаков).
- В операционной системе Linux и в Интернете часто используется кодировка UTF-8. В ней все символы, входящие в таблицу ASCII, кодируются с помощью 1 байта, а другие символы могут занимать от 2 до 6 байт.
- Для кодирования веб-страниц на русском языке используют кодировки UTF-8, Windows-1251 и KOI8-R.

Нарисуйте в тетради интеллект-карту этого параграфа.

Вопросы и задания

1. Вы хотите использовать в тексте придуманный собственный символ, которого нет ни в одном шрифте. Какими путями это можно сделать?
2. Вы сами разработали шрифт и хотите переслать другу документ, в котором этот шрифт используется. Какими способами это можно сделать?
3. В чём, по вашему мнению, достоинства и недостатки внедрения шрифтов в документ?
4. Почему в современных компьютерах используются кодировки, в которых каждый символ занимает целое число байт?
5. Почему использование кодовых страниц для кодирования текста может привести к проблемам?
6. В чём достоинства и недостатки использования кодировок UNICODE?
7. Что такое UTF-16 и UTF-8? Чем различаются эти кодировки?

§ 14
Кодирование графической информации

***Ключевые слова*:**

- пиксель
- разрешение
- цветовая модель RGB
- цветовая модель CMYK
- цветовая модель HSB
- глубина цвета
- цветовая палитра
- векторный рисунок
- кривые Безье
- 3D-графика
- рендеринг
- фрактальная графика

Изображения могут быть закодированы с помощью растрового и векторного методов.

При **растровом кодировании** рисунок разбивается на отдельные элементы (пиксели), каждый из которых закрашен одним цветом.

Пиксель — это наименьший элемент рисунка, для которого можно задать свой цвет.

Разрешение — это количество пикселей, приходящихся на единицу линейного размера изображения. Разрешение обычно измеряется в пикселях на дюйм (англ. **ppi**: *pixels per inch*).

При **векторном кодировании** рисунок представляется как множество фигур, контуры которых описываются математическими формулами.

Цветовые модели

Один из главных вопросов, которые возникают при кодировании изображений, — как хранить информацию о цвете?

Человек воспринимает свет как множество электромагнитных волн. Определённая длина волны соответствует некоторому цвету. Например, волны длиной 500–565 нм — это зелёный цвет, а волны длиной 625–740 нм — красный. Так называемый «белый» свет на самом деле представляет собой смесь волн, длины которых охватывают весь видимый диапазон.

Согласно современному представлению о **цветном зрении** (теории Юнга–Гельмгольца), глаз человека содержит чувствительные элементы («колбочки») трёх типов. Каждый из них воспринимает весь поток света, но первые наиболее чувствительны в области красного цвета, вторые — в области зелёного цвета, а третьи — в области синего цвета. Цвет — это результат возбуждения всех

трёх типов рецепторов (чувствительных элементов). Поэтому считается, что любой цвет (т. е. ощущения человека, воспринимающего волны определённой длины) можно имитировать, используя только три световых луча (красный, зелёный и синий) разной яркости. Следовательно, любой цвет (в том числе и «белый») приближённо раскладывается на три составляющих — красную, зелёную и синюю. Меняя силу этих составляющих, можно составить любые цвета. Эта модель цвета получила название **RGB** по начальным буквам английских слов *Red* (красный), *Green* (зелёный) и *Blue* (синий).

В **модели RGB** (рис. 2.18 и цветной рисунок на форзаце) яркость каждой составляющей (или, как говорят, каждого канала) чаще всего кодируется целым числом[1)] от 0 до 255. При этом код цвета — это тройка чисел (R, G, B), яркости отдельных каналов.

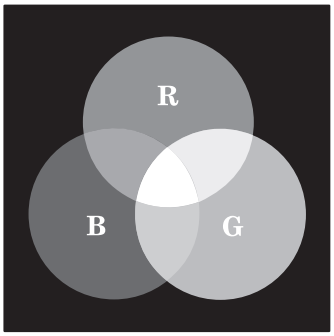


Рис. 2.18

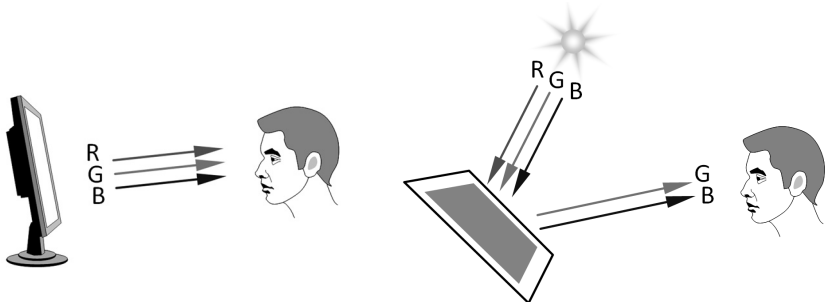
Цветовую модель RGB называют **аддитивной** (от англ. *add* — складывать), потому что нужный цвет получается «сложением» трёх базовых цветовых лучей. Так строится изображение на всех излучающих устройствах: экранах телевизоров, мониторах компьютеров и т. п. (рис. 2.19, *а*). Модель RGB также используется в сканерах и цифровых фотоаппаратах.

Цвет (0, 0, 0) в модели RGB — это чёрный цвет, а (255, 255, 255) — белый. Если все составляющие имеют равную яркость, получаются оттенки серого цвета, от чёрного до белого. Чтобы сделать светло-красный (розовый) цвет, нужно в красном цвете (255, 0, 0) одинаково увеличить яркость зелёного и синего каналов, например цвет (255, 150, 150) — это розовый. Равномерное уменьшение яркости всех каналов делает цвет темнее, например цвет с кодом (100, 0, 0) — тёмно-красный.

1) Или дробным числом от 0 до 1.

При кодировании цвета на веб-странице яркости каналов записывают в шестнадцатеричной системе счисления (от 00_{16} до FF_{16}), а перед кодом цвета ставится знак #. Например, код красного цвета записывается как #FF0000, а код синего — как #0000FF.

Когда мы смотрим на изображение, отпечатанное на бумаге (рис. 2.19, *б*), ситуация совершенно другая. Мы видим не прямые лучи источника, попадающие в глаз, а *отражённые* от поверхности.



а *б*

Рис. 2.19

«Белый свет» от какого-то источника (солнца, лампочки), содержащий волны во всём видимом диапазоне, попадает на бумагу, на которую нанесена краска. Краска поглощает часть лучей (их энергия уходит на нагрев), а оставшиеся попадают в глаз; это и есть тот цвет, который мы видим.

Например, если краска поглощает красные лучи, остаются только синие и зелёные — мы видим голубой цвет. В этом смысле красный и голубой цвета *дополняют* друг друга, так же как и пары «зелёный — пурпурный» и «синий — жёлтый». Действительно, если из белого цвета (его RGB-код #FFFFFF) «вычесть» зелёный, то получится цвет #FF00FF (пурпурный), а если «вычесть» синий, то получится цвет #FFFF00 (жёлтый).

На трёх *дополнительных цветах* — голубом, пурпурном и жёлтом — строится цветовая модель **CMY** (англ. *Cyan* — голубой, *Magenta* — пурпурный, *Yellow* — жёлтый), которая применяется для вывода на печать. Значения C = M = Y = 0 говорят о том, что на белую бумагу не наносится никакая краска, поэтому все лучи отражаются; это белый цвет. Если нанести на бумагу краску голубого цвета, красные лучи поглощаются, остаются только синие и зелёные (см. рис. 2.19, *б* и цветной рисунок на форзаце). Если сверху нанести ещё жёлтую краску, которая поглощает синие

лучи, остаётся только зелёный. Такая модель называется **субтрактивной** (от англ. *subtract* — вычитать) — рис. 2.20 и цветной рисунок на форзаце.

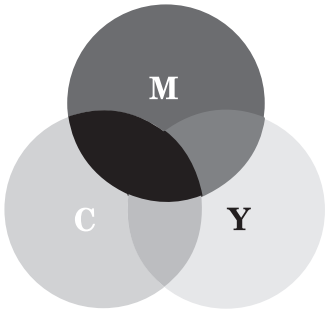


Рис. 2.20

При смешении голубой, пурпурной и жёлтой красок теоретически должен получиться чёрный цвет, все лучи поглощаются. Однако на практике всё не так просто. Краски не идеальны, поэтому вместо чёрного цвета получается грязно-коричневый. Кроме того, при печати чёрных областей приходится «выливать» тройную порцию краски в одно место. Нужно также учитывать, что обычно на принтерах часто распечатывают чёрный текст, а цветные чернила стоят значительно больше чёрных.

Чтобы решить эту проблему, в набор красок добавляют чёрную краску, это так называемый *ключевой цвет* (англ. *Key color*), поэтому получившуюся модель обозначают **CMYK**. Изображение, которое печатает большинство принтеров, состоит из точек этих четырёх цветов, которые расположены в виде узора очень близко друг к другу. Это создает иллюзию того, что в рисунке есть разные цвета.

Обычно изображения, предназначенные для печати, готовятся на компьютере (в режиме RGB), а потом переводятся в цветовую модель CMYK. При этом стоит задача получить при печати такой же цвет, что и на мониторе. И вот тут возникают проблемы. Дело в том, что при выводе пикселей на экран монитор получает некоторые числа (RGB-коды), на основании которых нужно «выкрасить» пиксели тем или иным цветом. Отсюда следует важный вывод.

Цвет, который мы видим на мониторе, зависит от характеристик и настроек монитора.

Это значит, что, например, красный цвет (R = 255, G = B = 0) на разных мониторах будет разным. Наверняка вы видели этот эффект в магазине, где продают телевизоры и мониторы — одна и та же картинка на каждом из них выглядит по-разному. Что же делать?

Во-первых, выполняется калибровка монитора — настройка яркости, контрастности, белого, чёрного и серого цветов. Во-вторых, профессионалы, работающие с цветными изображениями, используют *цветовые профили* мониторов, сканеров, принтеров и других устройств. В профилях хранится информация о том, каким реальным цветам соответствуют различные RGB-коды или CMYK-коды. Для создания профиля используют специальные приборы — *калибраторы* (*колориметры*), которые «измеряют» цвет с помощью трёх датчиков, принимающих лучи в красном, зелёном и синем диапазонах. Современные форматы графических файлов (например, формат PSD программы *Adobe Photoshop*) вместе с кодами пикселей хранят и профиль монитора, на котором создавался рисунок.

К сожалению, не все цвета RGB-модели могут быть напечатаны на бумаге. В первую очередь это относится к ярким и насыщенным цветам. Например, ярко-красный цвет (R = 255, G = B = 0) нельзя напечатать, ближайший к нему цвет в модели CMYK (C = 0, M = Y = 255, K = 0) при обратном переводе в RGB может дать значения[1] в районе R = 237, G = 28, B = 26. Поэтому при преобразовании ярких цветов в модель CMYK (и при печати ярких рисунков) они становятся тусклее. Это обязательно должны учитывать профессиональные дизайнеры.

Кроме цветовых моделей RGB и CMY (CMYK) существуют и другие. Наиболее интересная из них — модель **HSB**[2] (англ. ***H**ue* — тон, оттенок; ***S**aturation* — насыщенность, ***B**rightness* — яркость), которая ближе всего к естественному восприятию человека.

Тон (например, синий, зелёный или жёлтый) определяется длиной волны, он задаётся в градусах на цветовом круге (рис. 2.21 и цветной рисунок на форзаце). Насыщенность — это чистота тона, при уменьшении насыщенности до нуля получается серый цвет. Яркость определяет, насколько цвет яркий или тёмный. Любой цвет при снижении яркости до нуля превращается в чёрный. Чтобы получить белый цвет, нужно установить нулевую насыщенность и максимальную яркость, тон тут не важен.

---

1) Как вы понимаете, точные цифры зависят от профилей монитора и принтера.
2) Или **HSV** (англ. ***H**ue* — тон, оттенок; ***S**aturation* — насыщенность, ***V**alue* — величина).

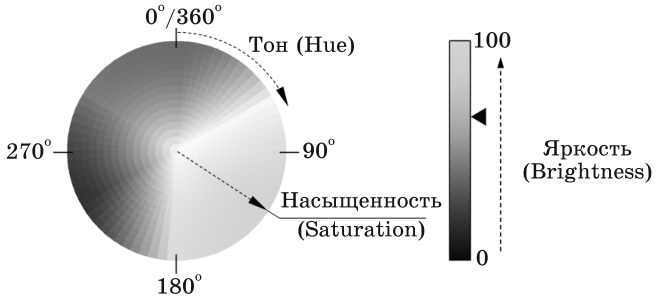


Рис. 2.21

Для кодирования «абсолютного» цвета, не зависящего от устройства, на котором он будет отображаться, применяют цветовую модель **Lab** (англ. ***Lightness*** — светлота, ***a*** и ***b*** — параметры, определяющие тон и насыщенность цвета), которая является международным стандартом. Эта модель используется, например, для перевода цвета из модели RGB в модель CMYK и обратно.

Для того чтобы результат печати на принтере был максимально похож на изображение на мониторе, нужно (используя профиль монитора) определить «абсолютный» цвет (например, в модели Lab), который видел пользователь, а потом (используя профиль принтера) найти CMYK-код, который даст при печати наиболее близкий цвет.

Растровое кодирование

Существуют два основных способа получения растровых изображений. При **вводе изображения** с помощью какого-либо устройства (сканера, цифрового фотоаппарата или веб-камеры) происходит **дискретизация** (оцифровка) — преобразование аналоговой информации в компьютерные данные. Как правило, оцифровка связана с потерей информации, поскольку требуется, чтобы каждый пиксель был залит одним цветом.

Второй способ — **создание рисунка** с помощью какой-либо компьютерной программы. В этом случае мы сразу строим цифровое изображение. Появилось даже отдельное направление в искусстве, которое называется «цифровая живопись» — создание электронных картин с помощью компьютерных технологий.

Пиксели растрового изображения обычно имеют квадратную форму, размеры рисунка (ширина и высота) также задаются в пикселях.

Для каждого пикселя в памяти хранится код его цвета. В модели RGB обычно используется по 256 вариантов яркости каждого из трёх цветов. Это позволяет закодировать $256^3 = 16\,777\,216$

оттенков, что более чем достаточно для человека. Так как $256 = 2^8$, каждая из трёх составляющих цвета занимает в памяти 8 бит (1 байт), а вся информация о каком-то цвете — 24 бита (3 байта).

Глубина цвета — это количество бит, используемых для кодирования цвета пикселя.

24-битное кодирование цвета часто называют режимом **истинного цвета** (англ. *True Color* — истинный цвет).

Задача 1. Определите информационный объём рисунка размером 20×30 пикселей, закодированного в режиме истинного цвета.

Решение. В режиме истинного цвета на каждый пиксель выделяется $i = 3$ байта. Количество пикселей изображения можно вычислить как произведение его размеров: $K = 20 \cdot 30 = 600$. Поэтому информационный объём рисунка равен $I = K \cdot i = 20 \cdot 30 \cdot 3$ байт = = 1800 байт.

Ответ: 1800 байт.

Конечно, при решении этой задачи мы не учитывали *сжатие* — уменьшение объёма файла с помощью специальных алгоритмов, — которое применяется во всех современных форматах графических файлов. Кроме того, в файлах всегда есть *заголовок*, в котором записана служебная информация (например, размеры рисунка).

Кроме режима истинного цвета используется также 16-битное кодирование (англ. *High Color* — «высокий» цвет), когда на красную и синюю составляющие отводится по 5 бит, а на зелёную, к которой человеческий глаз более чувствителен, — 6 бит. В режиме High Color можно закодировать $2^{16} = 65\,536$ различных цветов. Иногда применяют 12-битное кодирование цвета (4 бита на канал, 4096 цветов).

Как правило, чем меньше цветов используется, тем больше будет искажаться цвет. Таким образом, при кодировании цвета тоже возможна потеря информации, которая «добавляется» к потерям, вызванным дискретизацией. Однако при увеличении количества используемых цветов одновременно растёт объём файла. Например, в режиме истинного цвета файл получится в два раза больше, чем при 12-битном кодировании.

Если количество цветов в изображении невелико (не более 256), применяют **кодирование с палитрой**. **Цветовая палитра** — это таблица, в которой каждому цвету, заданному в виде составляющих в модели RGB, сопоставляется числовой код.

Палитра хранится в заголовке файла. Если рисунок содержит пиксели четырёх цветов — чёрного, красного, синего и белого, то его палитра содержит четыре трёхбайтных блока, в которых записаны RGB-коды этих цветов:

0	0	0	255	0	0	0	0	255	255	255	255
цвет $0 = 00_2$			цвет $1 = 01_2$			цвет $2 = 10_2$			цвет $3 = 11_2$		

Таким образом, размер палитры в байтах вычисляется как $3 \cdot N$, где N — количество цветов в палитре.

Для каждого пикселя изображения в памяти хранится не RGB-код цвета, а номер (код) соответствующего блока в палитре. Так как мы используем всего $2^2 = 4$ цвета, этот код занимает всего два бита, так что глубина цвета для такого рисунка составляет $i = 2$ бита на пиксель.

Задача 2. Какой минимальный объём памяти (в Кбайт) нужно зарезервировать, чтобы можно было сохранить любое растровое изображение размером 64×128 пикселей при условии, что в изображении могут использоваться 256 различных цветов?

Решение. Сначала находим количество пикселей K, перемножив ширину и высоту рисунка в пикселях. При этом удобно проводить все вычисления через степени числа 2:

$$K = 64 \cdot 128 = 2^6 \cdot 2^7 = 2^{13}.$$

Так как $256 = 2^8$, глубина цвета составляет $i = 8 = 2^3$ бит на пиксель. Поэтому информационный объём файла в битах равен $I = 2^{13} \cdot 2^3 = 2^{16}$ бит. Остаётся перевести его в килобайты:

$I = 2^{16} : 2^{13}$ Кбайт $= 2^3$ Кбайт $= 8$ Кбайт.

Ответ: 8 Кбайт.

Заметим, что мы вычислили только размер памяти, необходимый для хранения кодов пикселей (без учёта заголовка с палитрой).

Задача 3. Рисунок размером 512×256 пикселей занимает в памяти 64 Кбайт (без учёта сжатия). Найдите максимально возможное количество цветов в палитре изображения.

Решение. Максимально возможное количество цветов в палитре определяется глубиной цвета — количеством бит на пиксель. Чтобы найти глубину цвета, нужно разделить информационный объём изображения I на количество пикселей K.

Находим количество пикселей, перемножив размеры рисунка:

$$K = 512 \cdot 256 = 2^9 \cdot 2^8 = 2^{17}.$$

Информационный объём данных в битах равен

$$I = 64 \text{ Кбайт} = 2^6 \cdot 2^{13} \text{ бит} = 2^{19} \text{ бит}.$$

Теперь можно вычислить глубину цвета: $i = I : K = 219 : 217 = 4$ бита на пиксель. В этом случае максимально возможное количество цветов в палитре изображения равно $2^4 = 16$.

Ответ: 16 цветов.

Растровое кодирование — это универсальный метод, позволяющий кодировать любые изображения, в том числе размытые (например, фотографии). В то же время у него есть существенные **недостатки**:

- при дискретизации всегда есть *потеря информации*;
- при *изменении размеров* изображения искажается цвет и форма объектов на рисунке, поскольку при увеличении размеров надо как-то восстановить недостающие пиксели, а при уменьшении — заменить несколько пикселей одним;
- *размер файла* не зависит от сложности изображения, а определяется только разрешением и глубиной цвета; как правило, растровые рисунки имеют большой объём.

Форматы файлов

Существует много разных форматов хранения растровых рисунков. В большинстве из них используется сжатие, т. е. размер файла уменьшают с помощью специальных алгоритмов. В некоторых форматах применяют сжатие без потерь, при котором исходный рисунок можно в точности восстановить из сжатого состояния. Ещё бóльшую степень сжатия можно обеспечить, используя сжатие с потерями, при котором незначительная часть данных (почти не влияющая на восприятие рисунка человеком) теряется. Подробно мы изучим эти вопросы в 11 классе.

Чаще всего встречаются следующие форматы файлов:

- **BMP** (англ. *bitmap* — битовая карта, файлы с расширением *bmp*) — стандартный формат растровых изображений в операционной системе *Windows*; поддерживает кодирование с палитрой и в режиме истинного цвета;
- **JPEG** (англ. *Joint Photographic Experts Group* — объединённая группа фотографов-экспертов, файлы с расширением *jpg* или *jpeg*) — формат, разработанный специально для кодирования фотографий; поддерживает только режим истинного цвета; для уменьшения объёма файла используется сильное сжатие, при котором изображение немного размывается, поэтому не рекомендуется использовать его для рисунков с чёткими границами объектов;
- **GIF** (англ. *Graphics Interchange Format* — формат для обмена изображениями, файлы с расширением *gif*) — формат,

поддерживающий только кодирование с палитрой (от 2 до 256 цветов); в отличие от предыдущих форматов части рисунка могут быть прозрачными, т. е. на веб-странице через них будет «просвечивать» фон; в современном варианте формата GIF можно хранить анимированные изображения; используется сжатие без потерь;

- **PNG** (англ. *Portable Network Graphics* — переносимые сетевые изображения, файлы с расширением *png*) — формат, поддерживающий как режим истинного цвета, так и кодирование с палитрой; части изображения могут быть прозрачными и даже полупрозрачными (32-битное кодирование RGBA, где четвёртый байт задает прозрачность); изображение сжимается без искажения; анимация не поддерживается.

Свойства рассмотренных форматов сведены в таблицу:

Формат	Истинный цвет	С палитрой	Прозрачность	Анимация
BMP	Да	Да	—	—
JPEG	Да	—	—	—
GIF	—	Да	Да	Да
PNG	Да	Да	Да	—

Вы знаете, что все виды информации хранятся в памяти компьютера как двоичные коды, т. е. цепочки из нулей и единиц. Получив такую цепочку, абсолютно невозможно сказать, что это — текст, рисунок, звук или видео. Например, код 11001000_2 может обозначать число 200, букву «И», одну из составляющих цвета пикселя в режиме истинного цвета, номер цвета в палитре для рисунка с палитрой 256 цветов, цвета 8 пикселей чёрно-белого рисунка и т. п. Как же компьютер разбирается в двоичных данных? В первую очередь нужно ориентироваться на расширение имени файла. Например, чаще всего файлы с расширением *txt* содержат текст, а файлы с расширениями *bmp*, *gif*, *jpg*, *png* — рисунки.

Однако расширение файла можно менять как угодно. Например, можно сделать так, что текстовый файл будет иметь расширение *bmp*, а рисунок в формате JPEG — расширение *txt*. Поэтому в начало всех файлов специальных форматов (кроме простого текста, *txt*) записывается *заголовок*, по которому можно «узнать» тип файла и его характеристики. Например, файлы в формате BMP начинаются с символов «BM», а файлы в формате GIF — с символов «GIF». Кроме того, в заголовке указывается размер рисунка и его характеристики, например количество цветов в палитре, способ сжатия и т. п. Используя эту информацию,

программа «расшифровывает» основную часть файла и выводит данные на экран.

Векторное кодирование

Для чертежей, схем, карт применяется другой способ кодирования, который позволяет не терять качество при изменении размеров изображения.

Векторный рисунок — это рисунок, построенный из простейших геометрических фигур **(графических примитивов)**: линий, многоугольников, сглаженных кривых, окружностей, эллипсов. Их параметры (координаты вершин, цвет контура и заливки) хранятся в виде чисел.

Векторный рисунок можно «разобрать» на части, растащив мышью его элементы, а потом снова собрать полное изображение:

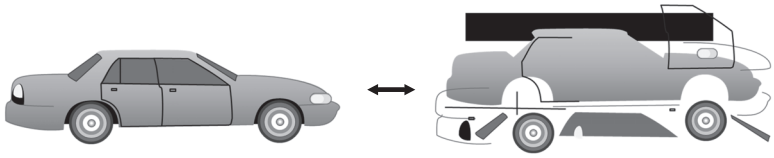


Рис. 2.22

Как вы понимаете, сделать что-то подобное с растровым рисунком не удастся.

При векторном кодировании для отрезка хранятся координаты его концов, для прямоугольников и ломаных — координаты вершин. Окружность и эллипс можно задать координатами прямоугольника, в который вписана фигура. Сложнее обстоит дело со сглаженными кривыми. На рисунке 2.23 изображена линия с опорными точками *А*, *Б*, *В*, *Г* и *Д*.

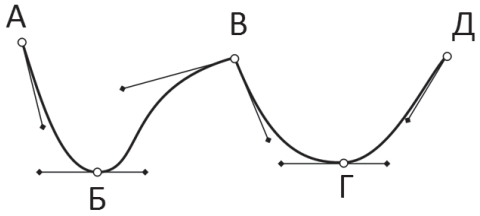


Рис. 2.23

У каждой из этих точек есть «рычаги» (*управляющие линии*), перемещая концы этих рычагов, можно регулировать наклон

касательной и кривизну всех участков кривой. Если оба рычага находятся на одной прямой, получается сглаженный узел (*Б* и *Г*), если нет — то угловой узел (*В*). Таким образом, форма этой кривой полностью задаётся координатами опорных точек и координатами рычагов. Кривые, заданные таким образом, называют *кривыми Безье* в честь их изобретателя французского инженера Пьера Безье.

Векторный способ кодирования рисунков обладает значительными **преимуществами** по сравнению с растровым:

- если изображение (например, чертёж, схема, карта, диаграмма) может быть полностью разложено на простейшие геометрические фигуры, то при кодировании нет потери информации;
- объём файлов напрямую зависит от сложности рисунка — чем меньше элементов, тем меньше места занимает файл. Как правило, векторные рисунки значительно меньше по объёму, чем растровые;
- при изменении размера векторного рисунка не происходит никакого искажения формы элементов, при увеличении наклонных линий не появляются «ступеньки», как при растровом кодировании (рис. 2.24).

Растровый рисунок Векторный рисунок

Рис. 2.24

Самый главный **недостаток** этого метода — он практически непригоден для кодирования изображений, в которых объекты не имеют чётких границ, например для фотографий.

Среди форматов векторных рисунков отметим следующие:

- **WMF** (англ. *Windows Metafile* — метафайл *Windows*, файлы с расширениями *wmf* и *emf*) — стандартный формат векторных рисунков в операционной системе *Windows*;
- **CDR** (файлы с расширением *cdr*) — формат векторных рисунков программы *CorelDRAW*;
- **AI** (файлы с расширением *ai*) — формат векторных рисунков программы *Adobe Illustrator*;

- **SVG** (англ. *Scalable Vector Graphics* — масштабируемые векторные изображения, файлы с расширением *svg*) — векторная графика для веб-страниц в Интернете.

Трёхмерная графика

Сейчас инженеры разрабатывают новые автомобили, самолёты, приборы в системах автоматизированного проектирования. В них строится трёхмерная (объёмная) модель объекта, которую затем можно просматривать с разных сторон и рассчитывать на прочность. Для создания объёмных моделей объектов (рис. 2.25) применяют **трёхмерную графику (3D-графику**, от англ. *3-dimensional* — трёхмерный).

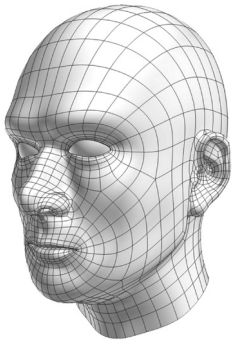


Рис. 2.25

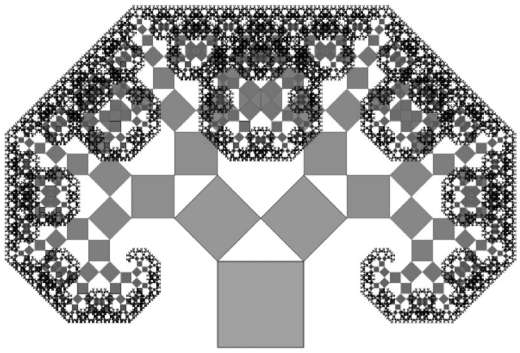
Трёхмерные модели хранятся в памяти компьютера как элементарные фигуры (отрезки, треугольники, четырёхугольники и др.) и поверхности, которые описываются математическими формулами. С этой точки зрения трёхмерная графика — это векторная графика.

Каркас объекта обычно строится из многоугольников (*полигонов*), потом его углы сглаживаются. На поверхности наносят рисунки, имитирующие реальные материалы, настраивают их свойства: цвет, прозрачность, блики и др. Затем устанавливают источники света, задают свойства атмосферы, в которой находится объект.

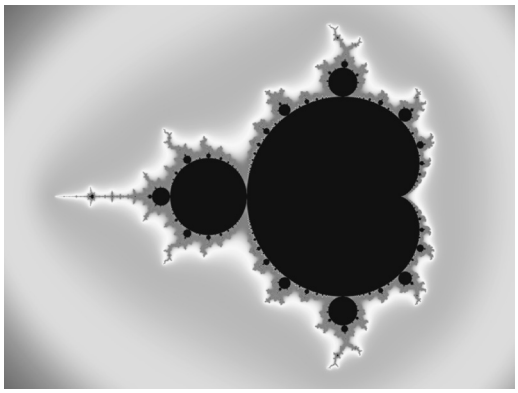
Для того чтобы построить двумерную картинку (проекцию трёхмерной модели на плоскость), нужно выбрать точку наблюдения («установить камеру») и просчитать, как выглядит модель с этой точки. Эта процедура называется *рендеринг*. Быстрая смена таких картинок позволяет строить анимацию — создавать иллюзию движения и изменения.

Фрактальная графика

Существует ещё один интересный вид графики — фрактальная графика. Слово «фрактал» образовано от латинского *fractus* и в переводе означает «состоящий из фрагментов». Фрактал обладает *самоподобием*, т. е. основная фигура состоит из нескольких таких же, только меньшего размера. Такие рисунки хранятся в памяти не как отдельные элементы (пиксели или графические примитивы), а в виде математической формулы и алгоритма построения. На рисунке 2.26 показаны два известных фрактала: дерево Пифагора (*а*) и множество Мандельброта (*б*) — см. цветной рисунок на форзаце.



а



б

Рис. 2.26

Фрактальная графика применяется для построения изображений растений, облаков, гор, водных поверхностей, а также для оформления рекламных листовок и веб-сайтов.

Выводы

- При растровом кодировании изображение разбивается на пиксели — элементы, для каждого из которых можно задать свой цвет независимо от других.
- Глубина цвета — это количество бит, используемых для кодирования цвета пикселя.
- Качество растрового кодирования зависит от разрешения и глубины цвета.
- Цвет пикселя при выводе на экран монитора кодируется в модели RGB — раскладывается на красную, зелёную и синюю составляющие.
- При выводе на печать используется цветовая модель CMYK (голубой, пурпурный, жёлтый, чёрный).
- При растровом кодировании, как правило, происходит потеря информации из-за дискретизации. Растровый рисунок искажается, когда его размеры изменяются.
- Векторный рисунок хранится в памяти как математическое описание множества геометрических фигур с заданными свойствами контура и заливки внутренней области.
- При изменении размеров векторного рисунка он не искажается.
- Трёхмерные модели (3D-модели) хранятся в памяти компьютера как элементарные фигуры и поверхности, которые описываются математическими формулами.
- Фрактал — это фигура, обладающая самоподобием, т. е. основная фигура состоит из нескольких таких же, только меньшего размера. Фракталы хранятся в памяти компьютера в виде математической формулы и алгоритма построения.

Нарисуйте в тетради интеллект-карту этого параграфа.

Вопросы и задания

1. Как вы думаете, почему не удаётся придумать единый метод кодирования рисунков, пригодный во всех ситуациях?
2. Как уменьшить потерю информации при дискретизации? Что при этом ухудшается?
3. Как связаны глубина цвета и объём файла?
4. Сравните режим истинного цвета и кодирование с палитрой.
5. Сравните растровой и векторный способы кодирования изображений.

6. В каких форматах целесообразно сохранять фотографии? Рисунки с чёткими границами объектов?
7. Как можно уменьшить объём файла, в котором хранится рисунок? Чем при этом придётся пожертвовать?
8. Как компьютер определяет, что находится в файле — текст, рисунок, звук или видео?
9. Узнайте, с каких символов начинается заголовок файла в формате PNG.

Подготовьте сообщение

а) «Цветовые модели XYZ и Lab»
б) «Фракталы»
в) «Сплайны»

Проекты

а) Сравнение форматов для хранения изображений
б) Программа для построения фракталов

§ 15 Кодирование звуковой и видеоинформации

Ключевые слова:

- оцифровка
- звуковая карта
- частота дискретизации
- разрядность кодирования
- кодек
- потоковый формат
- стандарт MIDI
- синтезатор
- сэмпл
- синхронность
- артефакты

Для кодирования звука используются два метода: оцифровка и инструментальное кодирование.

Оцифровка — это преобразование аналогового сигнала в цифровой код (последовательность чисел). При **инструментальном кодировании** в памяти компьютера хранится нотная запись мелодии и коды музыкальных инструментов.

Звук — это колебания среды (воздуха, воды). С помощью микрофона звук преобразуется в **аналоговый** электрический сигнал, который в любой момент времени может принимать любое значение в некотором интервале. Этот сигнал можно подать на вход звуковой карты, где специальное устройство — аналого-цифровой преобразователь (АЦП) — преобразует его в цифровой код. Процессор компьютера может затем обработать этот код по некоторому алгоритму, сохранить в файле и т. д. (рис. 2.27).

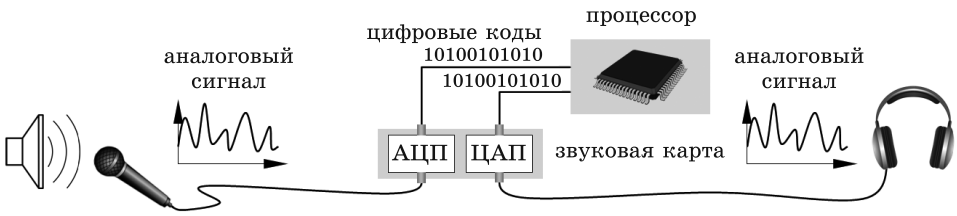


Рис. 2.27

Для проигрывания звука через наушники или звуковые колонки (это аналоговые устройства!), цифровой код из памяти компьютера (например, из файла) передаётся звуковой карте, где с помощью цифро-аналогового преобразователя (ЦАП) преобразуется в аналоговый сигнал, поступающий на устройство вывода звука.

Оцифровка звука

При оцифровке звука выполняется **дискретизация** — из всего бесконечного множества значений аналогового сигнала сохраняются в памяти только значения в отдельных точках, взятых с некоторым шагом T по времени (рис. 2.28, *а*). Это называется **дискретизацией по времени**.

Число T называется **интервалом дискретизации**, а обратная ему величина $f = 1/T$ — **частотой дискретизации**. Частота дискретизации измеряется в герцах (Гц) и килогерцах (кГц). Чем больше частота дискретизации, тем точнее мы записываем сигнал, тем меньше информации теряем. Однако при этом возрастает количество отсчётов, т. е. информационный объём закодированного звука. Для кодирования звука в компьютерах чаще всего используются частоты дискретизации 8 кГц (минимальное качество, достаточное для распознавания речи), 11 кГц, 22 кГц, 44,1 кГц (звуковые компакт-диски), 48 кГц (фильмы в формате DVD), а также 96 кГц и 192 кГц (высококачественный звук в формате DVD-audio).

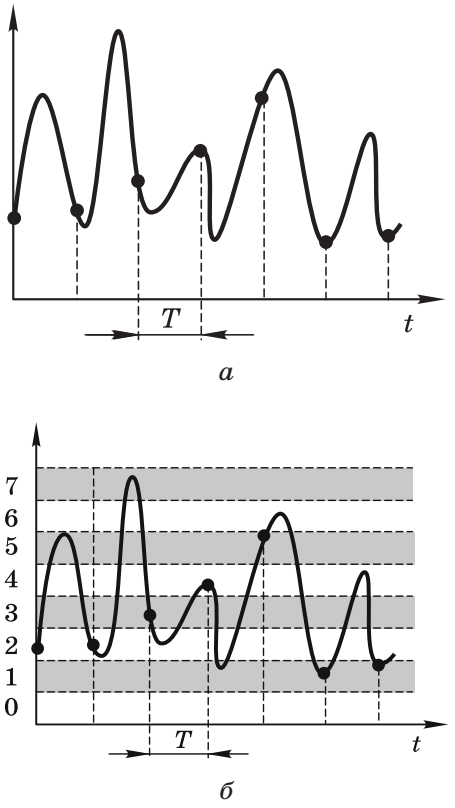


Рис. 2.28

Кроме дискретизации по времени в АЦП происходит и **дискретизация по уровню** (квантование): измеренные значения сигнала записываются в памяти как целые числа. На рис. 2.28, *б* весь диапазон значений сигнала разбит на $8 = 2^3$ одинаковых полос, что соответствует 3-битному кодированию. Все значения, попавшие в одну полосу, получают одинаковые коды.

Разрядность кодирования (глубина кодирования) — это число бит, используемое для хранения одного отсчёта.

Недорогие звуковые карты имеют разрядность 16–18 бит, большинство современных — 24 бита, что позволяет использовать $2^{24} = 16\,777\,216$ различных уровней.

Информационный объём данных, полученных в результате оцифровки звука, равен

$$I = f \cdot i \cdot t \cdot k,$$

где f — частота квантования, i — разрядность кодирования, t — время и k — число каналов, которые записываются одновременно.

Для стереофонической записи (когда отдельно записываются левый и правый каналы) нужно принять $k = 2$, а для квадрофонического звука (запись четырёх каналов одновременно) — $k = 4$.

Например, если используется 16-разрядное кодирование с частотой 44 кГц, то за 1 с выполняется 44 000 измерений сигнала, и каждое из измеренных значений занимает 16 бит (2 байта). Поэтому за 1 секунду накапливается $f \cdot i = 44000 \cdot 2 = 88\,000$ байт данных, а за 1 минуту

$$f \cdot i \cdot t = 88\,000 \cdot 60 = 5\,280\,000 \text{ байт} \approx 5 \text{ Мбайт}.$$

Если записывается стереозвук, это число нужно удвоить, а при записи квадрофонического звука —умножить на четыре.

Восстановление звукового сигнала

При проигрывании звука приходится решать сложную задачу — восстанавливать аналоговый сигнал по его дискретным значениям, взятым с некоторой частой f. С точки зрения математики, любой сигнал можно представить в виде суммы очень большого числа колебаний разных частот (гармоник). Если выбрать частоту дискретизации f больше, чем удвоенная частота самой быстрой гармоники, то теоретически по отдельным отсчётам можно точно восстановить исходный аналоговый сигнал. Это результат известен в радиотехнике как теорема Котельникова–Шеннона.

К сожалению, на практике всё несколько сложнее. Дело в том, что в реальных сигналах содержатся гармоники с очень высокими частотами, так что частота дискретизации, полученная с помощью теоремы Котельникова–Шеннона, будет также высока, и объём файла недопустимо велик.

Однако средний человек слышит только звуки с частотами от 16 Гц до 20 кГц, поэтому все частоты выше 20 кГц можно «потерять» практически без ухудшения качества звука (человек не почувствует разницу!). Удвоив эту частоту (по теореме Котельникова–Шеннона), получаем оптимальную частоту

дискретизации около 40 кГц, которая обеспечивает наилучшее качество, различимое на слух. Поэтому при высококачественном цифровом кодировании звука на компакт-дисках и в видеофильмах чаще всего используют частоты 44,1 кГц и 48 кГц. Более низкие частоты дискретизации применяют тогда, когда важно всячески уменьшать объём звуковых данных (например, для трансляции радиопередач через Интернет), даже ценой ухудшения качества.

Простейший метод восстановления сигнала по отдельным отсчётам — построить ступенчатый сигнал (рис. 2.29). В современных звуковых картах для повышения качества звука этот ступенчатый сигнал сглаживается с помощью специальных фильтров, однако восстановить точно исходный сигнал всё равно не удаётся, так как информация о значениях сигнала между моментами дискретизации была потеряна при оцифровке.

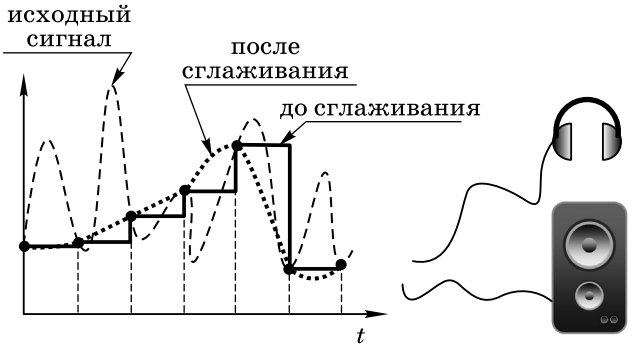


Рис. 2.29

С помощью оцифровки можно закодировать любой звук, который принимает микрофон. Однако при оцифровке звука всегда есть потеря информации (из-за дискретизации). Кроме того, звуковые файлы имеют, как правило, большой размер, поэтому в большинстве современных форматов используется сжатие; программа, которая выполняет такое сжатие, называется **кодеком** (от англ. *coder/decoder* — кодировщик/декодировщик).

Среди форматов оцифрованных звуковых файлов наиболее известны форматы:

- **WAV** (англ. *Waveform Audio File Format*, файлы с расширением *wav*);
- **MP3** (файлы с расширением *mp3*);

- **AAC** (англ. *Advanced Audio Coding*, файлы с расширениями *aac*, *mp4*, *m4a* и др.);
- **WMA** (англ. *Windows Media Audio*, файлы с расширением *wma*);
- **Ogg Vorbis** (файлы с расширением *ogg*) — открытый формат, не требующий оплаты лицензии.

Все эти форматы — **потоковые**, т. е. можно начинать прослушивание звука до того момента, как весь файл будет получен (например, из Интернета). Как правило, в них используется *сжатие с потерями*: для значительного уменьшения объёма файла снижается качество кодирования для тех частот, которые практически неразличимы для человеческого слуха.

Инструментальное кодирование звука

Для кодирования инструментальных мелодий нередко используется стандарт **MIDI** (англ. *Musical Instrument Digital Interface* — цифровой интерфейс музыкальных инструментов). В отличие от оцифрованного звука в таком формате хранятся последовательность нот, коды инструментов (можно использовать 128 мелодических и 47 ударных инструментов), громкость, тембр, время затухания каждой ноты и т. д. Фактически это программа, предназначенная для проигрывания звуковой картой, в памяти которой хранятся образцы звуков реальных инструментов (волновые таблицы, англ. *wave tables*).

Современные звуковые карты поддерживают многоканальный звук, т. е. в звуковом файле может храниться несколько «дорожек», которые проигрываются одновременно. Таким образом, получается *полифония* — многоголосие, возможность проигрывать одновременно несколько нот. Количество голосов для современных звуковых карт может достигать 1024.

Звук, закодированный с помощью стандарта MIDI, хранится в файлах с расширением *mid*. Для проигрывания MIDI-файла используют *синтезаторы* — электронные устройства, имитирующие звук реальных инструментов. Простейший синтезатор — звуковая карта компьютера.

Главные **достоинства** инструментального кодирования:

- кодирование мелодии (нотной записи) происходит без потери информации;
- файлы имеют значительно меньший объём в сравнении с оцифрованным звуком той же длительности.

Однако произвольный звук (например, человеческий голос) в таком формате закодировать невозможно. Кроме того, производители сами выбирают образцы звуков (так называемые *сэмплы*, от англ. *samples* — образцы), которые записываются в память звуковой карты (нет единого стандарта). Поэтому звучание MIDI-файла может немного отличаться на разной аппаратуре.

Кодирование видеоинформации

Для того чтобы сохранить видео в памяти компьютера, нужно закодировать звук и изменяющееся изображение, причём при проигрывании требуется обеспечить их *синхронность* (одновременность).

Для кодирования звука чаще всего используют оцифровку с частотой 48 кГц. Изображение состоит из отдельных растровых рисунков, которые меняются с частотой не менее 25 кадров в секунду, так что глаз человека воспринимает смену кадров как непрерывное движение. Это значит, что для каждой секунды видео нужно хранить в памяти 25 изображений.

При размере кадра 768×576 точек и глубине цвета 24 бита на пиксель закодированная 1 секунда видео (без звука) будет занимать примерно 32 Мбайт, а 1 минута — около 1,85 Гбайт. Это недопустимо много, поэтому в большинстве форматов видеоизображений используется сжатие. Упаковку и распаковку видеоданных выполняют программы-**кодеки**.

Основная идея сжатия видеофайлов основана на том, что за короткое время изображение изменяется очень мало, поэтому можно запомнить «базовый» кадр, а затем сохранять только изменения. Через 10–15 с изображение изменяется настолько, что необходим новый базовый кадр. Для того чтобы ещё больше уменьшить объём файла, применяют сжатие с потерями, при котором теряются некоторые детали, несущественные для восприятия человеком. При очень сильном сжатии с потерями появляются заметные искажения **(артефакты)**, например становится явно видно, что изображение разбито на блоки размером 8×8 пикселей.

Современные цифровые видеокамеры и фотоаппараты могут записывать видео в форматах высокой чёткости с размерами изображения 1280×720 пикселей и 1920×1080 пикселей (*Full HD*).

Среди форматов видеофайлов наиболее известны:

- **AVI** (англ. *Audio Video Interleave* — чередующиеся звук и видео, файлы с расширением *avi*);

- **WMV** (англ. *Windows Media Video*, файлы с расширением *wmv*);
- **MPEG** (файлы с расширением *mpg*, *mpeg*);
- **MP4** (файлы с расширением *mp4*);
- **MOV** (англ. *Quick Time Movie*, файлы с расширением *mov*);
- **WebM** — открытый (не требующий оплаты лицензии) видеоформат, который поддерживается в современных браузерах без установки дополнительных модулей.

Выводы

- Для кодирования звука применяют оцифровку и инструментальное кодирование.
- Оцифровка — это дискретизация аналогового сигнала: в памяти сохраняются значения этого сигнала, измеряемые с некоторой частотой, которая называется частотой дискретизации.
- Качество оцифровки определяется частотой дискретизации и разрядностью кодирования (количеством бит, выделяемых для хранения одного измеренного значения).
- Инструментальное кодирование звука позволяет без искажений кодировать нотную запись, но его нельзя использовать для кодирования нестандартных звуков, например речи человека.
- Кодирование видеофильмов сводится к кодированию изменяющегося изображения и звуковых дорожек. Изображение и звук должны проигрываться одновременно (синхронно).
- Для уменьшения объёма файлов при кодировании звука и видеоданных, как правило, используют сжатие с потерями.

Нарисуйте в тетради интеллект-карту этого параграфа.

Вопросы и задания

1. Сравните оцифровку звука и инструментальное кодирование. Как вы думаете, почему не удаётся придумать один способ кодирования звука, который был бы лучше других во всех случаях?
2. Как связаны частота дискретизации с потерей информации и объёмом файла?
3. От чего зависит выбор частоты дискретизации? Почему частоты дискретизации более 48 кГц применяются очень редко?

4. Какие проблемы возникают при проигрывании цифрового звука?
5. Как связана разрядность кодирования звука с качеством закодированного звука и размером файла?
6. Почему мелодии, закодированные по стандарту MIDI, могут по-разному звучать на разной аппаратуре?
7. По какому принципу выбирается частота смены кадров и частота дискретизации звука при кодировании видеофильмов?
8. Какие принципы используются для сжатия видеофильмов?

Подготовьте сообщение

а) «Программы для создания MIDI-музыки»
б) «Кодирование звука в формате MP3»

Проекты

а) Сравнение форматов для хранения звука
б) Сравнение видеоформатов

ЭОР на сайте ФЦИОР (http://fcior.edu.ru)

- Представление текста в различных кодировках
- Принцип дискретного (цифрового) представления информации, системы счисления, алгоритмы
- Достоинства и недостатки двоичной системы счисления при использовании её в компьютере
- Понятие о системах счисления
- Представление числовой информации с помощью систем счисления. Алфавит, базис, основание. Свёрнутая и развёрнутая формы представления чисел
- Алгоритм перевода дробных чисел из 10-й системы счисления в *P*-ичную
- Алгоритм перевода целых чисел из 10-й системы счисления в *P*-ичную
- Арифметические операции в позиционных системах счисления
- Связь между двоичной, восьмеричной и шестнадцатеричной системами счисления
- Алгоритм перевода целых чисел из *P*-ичной системы счисления в 10-ю

- Представление текста в различных кодировках
- Аппаратное и программное обеспечение для представления изображения
- Растровая и векторная графика
- Аппаратное и программное обеспечение для представления изображения
- Аппаратное и программное обеспечение для представления звука

Практические работы к главе 2

www

Работа № 5 «Декодирование»
Работа № 6 «Необычные системы счисления»