В. В. Лидовский

ТЕОРИЯ ИНФОРМАЦИИ

В. В. ЛИДОВСКИЙ

ТЕОРИЯ ИНФОРМАЦИИ

МОСКВА
2003

Лидовский В. В. **Теория информации**: Учебное пособие. — М.: ?????????, 2003. — 112 с. — ISBN ?-????-????-?.

В учебном пособии излагаются основные понятия и факты теории информации. Рассмотрены способы измерения, передачи и обработки информации. Значительное внимание уделено свойствам меры информации, характеристикам канала связи, помехозащитному, уплотняющему и криптографическому кодированию. Кроме того, рассмотрены вопросы формализации информации, в частности, в документах Internet. Изложение сопровождается большим количеством примеров и упражнений.

Для студентов втузов соответствующих специальностей и всех интересующихся вопросами точной работы с информацией и методами построения кодов с полезными свойствами.

Библиогр. 23 назв. Ил. 28.

РЕЦЕНЗЕНТЫ: Кафедра “Управления и моделирования систем” Московской государственной академии приборостроения и информатики (зав. кафедрой — д-р. тех. наук С. Н. Музыкин), доцент *Н. Я. Смирнов*

Для подготовки издания использовались системы
plain TEX, $\mathcal{AMS}$-Fonts, PICTEX и TreeTEX

Введение

Учебное пособие написано на основе односеместрового 108 часового курса лекций и материалов для практических занятий, используемых автором в учебной работе со студентами-третьекурсниками в течении 5 лет на кафедре “Моделирование систем и информационные технологии” «МАТИ» — Российского государственного технологического университета им. К. Э. Циолковского.

Настоящее пособие достаточно полно освещает основные положения теории информации в соответствии с Государственным образовательным стандартом РФ от 1995 г. по специальности “Автоматизированные системы обработки информации и управления” (220200). Содержание некоторых глав (2, 9, 33–36) пособия выходит за рамки стандарта для означенной специальности, но затронутые в них темы актуальны и органично вписываются в материал пособия.

Содержание пособия во многом базируется на некоторых вводных понятиях курса “Теория вероятностей”: дискретная случайная величина (д.с.в.), закон распределения вероятностей, математическое ожидание (м.о.) и т.п. Кроме того, от читателя требуется умение выполнять соответствующие операции с матрицами, многочленами и булевыми величинами.

В главах с 1 по 9 рассмотрены общие вопросы, определяющие практические подходы к использованию понятия *информация*, т.е. дано определение основных терминов, используемых при работе с информацией, очерчен круг вопросов, рассматриваемых в теории информации, приведены способы хранения, обработки, преобразования, передачи и измерения информации.

В главах 10–18 рассматриваются способы сжатия информации. Рассмотрены как статистические методы (Шеннона-Фэно, Хаффмена, арифметический), так и словарные методы Лемпела-Зива. Для статистических методов приведены варианты адаптивных алгоритмов кодирования. Приводятся формулы для оценки предельной степени сжатия информации. Обзорно рассматриваются способы сжатия информации с потерями и типы файлов, содержащих сжатые данные.

Глава 19 посвящена физическому уровню передачи информации по каналам связи. Рассматриваются методы расчета пропускной способности (емкости) канала, теорема Шеннона и обратная ей теорема, способы кодирования дискретной информации для передачи. Полное раскрытие названных тем требует привлечения мощного аппарата средств теории вероятностей и теории связи, выходящих за рамки соответствующих курсов студентов втузов, поэтому эти темы раскрыты лишь частично, в обзорном порядке.

В главах 20–27 рассматриваются способы построения и использо-

вания избыточных кодов для защиты от помех. Приводятся фундаментальные характеристики таких кодов. Для понимания материала 23-й главы необходимо знакомство с начальными элементами теории групп.

Главы 28–32 посвящены вопросам теории защиты информации. Рассматриваются как классические криптографические системы, так и системы, построенные на идеях Диффи и Хеллмана. Кратко математический фундамент этих методов излагается в Приложении Д.

В заключительных главах рассмотрены некоторые вопросы использования информации в Internet.

Используемые обозначения, не определенные явно в основном материале, приводятся в Приложении Е.

Ссылки на литературу из Приложения Ж, содержащую обоснования приведенных фактов или дополнительные подробности, заключаются в квадратные скобки.

Высокая требовательность студенческой аудитории является постоянным стимулом в поиске более простых, доходчивых и ясных способов изложения. Автор надеется, что это учебное пособие, формировавшееся в процессе живого общения со студентами, не получилось чрезмерно сложным.

Автор считает необходимым выразить искреннюю благодарность всем тем, кто помог ему в создании этого пособия, в особенности, Пантелееву П. А., Лидовской В. В. и Бурашникову С. Р.

1. **Предмет и основные разделы кибернетики**

Теория информации рассматривается как существенная часть кибернетики.

Кибернетика — это наука об общих законах получения, хранения, передачи и переработки информации. Ее основной предмет исследования — это так называемые кибернетические системы, рассматриваемые абстрактно, вне зависимости от их материальной природы. Примеры кибернетических систем: автоматические регуляторы в технике, ЭВМ, мозг человека или животных, биологическая популяция, социум. Часто кибернетику связывают с методами искусственного интеллекта, т. к. она разрабатывает общие принципы создания систем управления и систем для автоматизации умственного труда. Основными разделами (они фактически абсолютно самостоятельны и независимы) современной кибернетики считаются: теория информации, теория алгоритмов, теория автоматов, исследование операций, теория оптимального управления и теория распознавания образов.

Родоначальниками кибернетики (датой ее рождения считается 1948 год, год соответствующей публикации) считаются американские ученые Норберт Винер (Wiener, он — прежде всего) и Клод Шеннон (Shannon, он же основоположник теории информации).

Винер ввел основную категорию кибернетики — *управление*, показал существенные отличия этой категории от других, например, энергии, описал несколько задач, типичных для кибернетики, и привлек всеобщее внимание к особой роли вычислительных машин, считая их индикатором наступления новой НТР. Выделение категории управления позволило Винеру воспользоваться понятием информации, положив в основу кибернетики изучение законов передачи и преобразования информации.

Сущность принципа управления заключается в том, что движение и действие больших масс или передача и преобразование больших количеств энергии направляется и контролируется при помощи небольших количеств энергии, несущих информацию. Этот принцип управления лежит в основе организации и действия любых управляемых систем: автоматических машин и живых организмов. Подобно тому, как введение понятия энергии позволило рассматривать все явления природы с единой точки зрения и отбросило целый ряд ложных теорий, так и введение понятия информации позволяет подойти с единой точки зрения к изучению самых различных процессов взаимодействия в природе.

В СССР значительный вклад в развитие кибернетики внесли академики Берг А. И. и Глушков В. М.

В нашей стране в 50-е годы кибернетика была объявлена лженаукой и была практически запрещена, что не мешало, однако, развиваться всем ее важным разделам (в том числе и теории информации) вне связи с обобщающим словом “кибернетика”. Это было связано с тем, что сама по себе кибернетика представляет собой род философии, в коем-чем конфликтной с тогдашней официальной доктриной (марксистско-ленинской диалектикой).

Теория информации тесно связана с такими разделами математики как теория вероятностей и математическая статистика, а также прикладная алгебра, которые предоставляют для нее математический фундамент. С другой стороны теория информации исторически и практически представляет собой математический фундамент теории связи. Часто теорию информации вообще рассматривают как одну из ветвей теории вероятностей или как часть теории связи. Таким образом, предмет “Теория информации” весьма узок, т. к. зажат между “чистой” математикой и прикладными (техническими) аспектами теории связи.

Теория информации представляет собой математическую теорию, посвященную измерению информации, ее потока, “размеров” канала связи и т. п., особенно применительно к радио, телеграфии, телевидению и к другим средствам связи. Первоначально теория была посвящена каналу связи, определяемому длиной волны и частотой, реализация которого была связана с колебаниями воздуха или электромагнитным излучением. Обычно соответствующий процесс был непрерывным, но

мог быть и дискретным, когда информация кодировалась, а затем декодировалась. Кроме того, теория информации изучает методы построения кодов, обладающих полезными свойствами.

2. **Формальное представление знаний**

При формальном представлении знаний каждому описываемому объекту или понятию ставится в соответствие некоторый числовой код. Связи между кодируемыми сущностями также представляются кодами (адресами и указателями). Для такого перевода неформальных данных в формальный, цифровой вид должны использоваться специальные таблицы, сопоставляющие кодируемым сущностям их коды и называемые *таблицами кодировки*. Простейший пример такой таблицы — это ASCII (American Standard Code for Information Interchange), используемая повсеместно с вычислительной техникой. Она сопоставляет печатным и управляющим символам (управляющими являются, например, символы, отмечающие конец строки или страницы) числа от 0 до 127. Следующая программа на языке Паскаль выведет на экран все печатные символы этой таблицы и их коды:

```
var i: byte;
begin
  for i := 32 to 126 do
    write(i:6, chr(i):2);
  writeln
end.
```

На практике обычно используют не сам исходный ASCII, а так называемый расширенный ASCII (ASCII+), описывающий коды 256 символов (от 0 до 255). Первые 128 позиций расширенного ASCII совпадают со стандартом, а дополнительные 128 позиций определяются производителем оборудования или системного программного обеспечения. Кроме того, некоторым управляющим символам ASCII иногда назначают другое значение.

Хотя таблицы кодировки используются для формализации информации, сами они имеют неформальную природу, являясь мостом между реальными и формальными данными. Например, коду 65 в ASCII соответствует заглавная латинская буква A, но не конкретная, а любая. Этому коду будет соответствовать буква **A**, набранная жирным прямым шрифтом, и буква *A*, набранная нежирным с наклоном вправо на 9.5° шрифтом, и даже буква $\mathfrak{A}$ готического шрифта. Задача сопоставления реальной букве ее кода в выбранной таблице кодировки очень сложна и частично решается программами распознания символов (например, Fine Reader).

▶ **Упражнение 1**
Каков код букв W и w в ASCII?

3. **Виды информации**

Информация может быть двух видов: *дискретная* (*цифровая*) и *непрерывная* (*аналоговая*). Дискретная информация характеризуется последовательными точными значениями некоторой величины, а непрерывная — непрерывным процессом изменения некоторой величины. Непрерывную информацию может, например, выдавать датчик атмосферного давления или датчик скорости автомашины. Дискретную информацию можно получить от любого цифрового индикатора: электронных часов, счетчика магнитофона и т. п.

Дискретная информация удобнее для обработки человеком, но непрерывная информация часто встречается в практической работе, поэтому необходимо уметь переводить непрерывную информацию в дискретную (дискретизация) и наоборот. Модем (это слово происходит от слов модуляция и демодуляция) представляет собой устройство для такого перевода: он переводит цифровые данные от компьютера в звук или электромагнитные колебания-копии звука и наоборот.

При переводе непрерывной информации в дискретную важна так называемая *частота дискретизации* ν, определяющая период ($T = 1/\nu$) определения значения непрерывной величины (см. рис. 1).

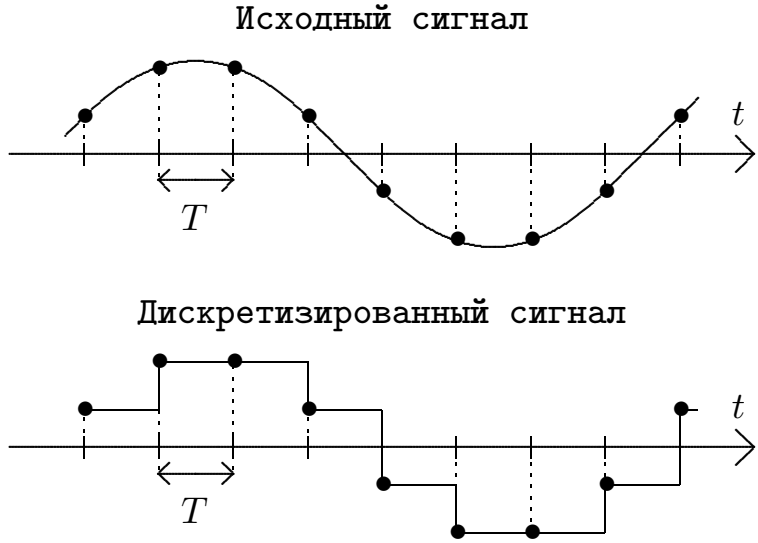


Рис. 1

Чем выше частота дискретизации, тем точнее происходит перевод непрерывной информации в дискретную. Но с ростом этой частоты растет и размер дискретных данных, получаемых при таком переводе, и, следовательно, сложность их обработки, передачи и хранения. Однако для повышения точности дискретизации необязательно безграничное увеличение ее частоты. Эту частоту разумно увеличивать только до

предела, определяемого теоремой о выборках или законом Найквиста (Nyquist).

Любая непрерывная величина описывается множеством наложенных друг на друга волновых процессов, называемых гармониками, определяемых функциями вида $A\sin(\omega t + \varphi)$, где A — это амплитуда, ω — частота, t — время и φ — фаза.

Теорема о выборках гласит, что для точной дискретизации ее частота должна быть не менее чем в два разы выше наибольшей частоты гармоники, входящей в дискретизируемую величину [17].

Примером использования этой теоремы являются лазерные компакт-диски, звуковая информация на которых хранится в цифровой форме. Чем выше будет частота дискретизации, тем точнее будут воспроизводиться звуки и тем меньше их можно будет записать на один диск, но ухо обычного человека способно различать звуки с частотой до 20 КГц, поэтому точно записывать звуки с большей частотой бессмысленно. Согласно теореме о выборках частоту дискретизации нужно выбрать не меньшей 40 КГц (в промышленном стандарте на компакт-диске используется частота 44.1 КГц).

При преобразовании дискретной информации в непрерывную, определяющей является скорость этого преобразования: чем она выше, с тем более высокочастотными гармониками получится непрерывная величина. Но чем большие частоты встречаются в этой величине, тем сложнее с ней работать. Например, обычные телефонные линии предназначены для передачи звуков частотой до 3 КГц. Связь скорости передачи и наибольшей допустимой частоты подробнее будет рассмотрена далее.

Устройства для преобразования непрерывной информации в дискретную обобщающе называются АЦП (аналого-цифровой преобразователь) или ADC (Analog to Digital Convertor, A/D), а устройства для преобразования дискретной информации в аналоговую — ЦАП (цифро-аналоговый преобразователь) или DAC (Digital to Analog Convertor, D/A).

► Упражнение 2
В цифровых магнитофонах DAT частота дискретизации — 48 КГц. Какова максимальная частота звуковых волн, которые можно точно воспроизводить на таких магнитофонах?

4. Хранение, измерение, обработка и передача информации

Для хранения информации используются специальные устройства памяти. Дискретную информацию хранить гораздо проще непрерывной, т. к. она описывается последовательностью чисел. Если представить каждое число в двоичной системе счисления, то дискретная информация предстанет в виде последовательностей нулей и единиц. При-

сутствие или отсутствие какого-либо признака в некотором устройстве может описывать некоторую цифру в какой-нибудь из этих последовательностей. Например, позиция на дискете описывает место цифры, а полярность намагниченности — ее значение. Для записи дискретной информации можно использовать ряд переключателей, перфокарты, перфоленты, различные виды магнитных и лазерных дисков, электронные триггеры и т. п. Одна позиция для двоичной цифры в описании дискретной информации называется *битом* (bit, binary digit). Бит служит для измерения информации. Информация размером в один бит содержится в ответе на вопрос, требующий ответа “да” или “нет”.

Хранить непрерывную информацию очень сложно. Обычно для этого используют электрические схемы на основе конденсатора. Непрерывную информацию тоже измеряют в битах.

Бит — это очень маленькая единица, поэтому часто используется величина в 8 раз большая — *байт (byte)*, состоящая из двух 4-битных полубайт или тетрад. Байт обычно обозначают заглавной буквой В или Б. Как и для прочих стандартных единиц измерения для бита и байта существуют производные от них единицы, образуемые при помощи приставок кило (K), мега (M), гига (G или Г), тера (T), пета (P или П) и других. Но для битов и байтов они означают не степени 10, а степени двойки: кило — $2^{10} = 1024 \approx 10^3$, мега — $2^{20} \approx 10^6$, гига — $2^{30} \approx 10^9$, тера — $2^{40} \approx 10^{12}$, пета — $2^{50} \approx 10^{15}$. Например, 1 КВ = 8 Kbit = 1024 В = 8192 bit, 1 МБ = 1024 КБ = 1 048 576 Б = 8192 Кбит.

Для обработки информации используют вычислительные машины, которые бывают двух видов: ЦВМ (цифровая вычислительная машина) — для обработки дискретной информации, АВМ (аналоговая вычислительная машина) — для обработки непрерывной информации. ЦВМ — универсальны, на них можно решать любые вычислительные задачи с любой точностью, но с ростом точности скорость их работы уменьшается. ЦВМ — это обычные компьютеры.

Каждая АВМ предназначена только для узкого класса задач, например, интегрирования или дифференцирования. Если на вход такой АВМ подать сигнал, описываемый функцией $f(t)$, то на ее выходе появится сигнал $F(t)$ или $f'(t)$. АВМ работают очень быстро, но их точность ограничена и не может быть увеличена без аппаратных переделок. Программа для АВМ — это электрическая схема из заданного набора электронных компонент, которую нужно физически собрать.

Бывают еще и гибридные вычислительные машины, сочетающие в себе элементы как ЦВМ, так и АВМ.

На рис. 2 изображена схема передачи информации.

Кодированием, например, является шифровка сообщения, декодированием — его дешифровка.

Процедуры кодирования и декодирования могут повторяться мно-

го раз. Ошибки при передаче информации происходят из-за: шума в канале (атмосферные и технические помехи), ошибки кодирования и декодирования. Теория информации изучает, в частности, способы минимизации количества таких ошибок.

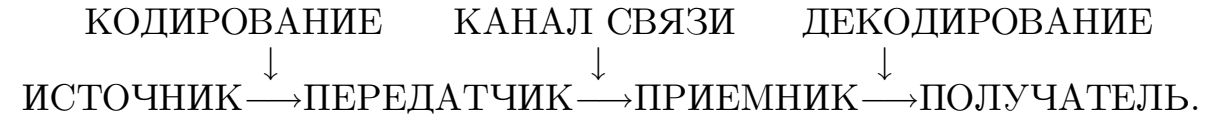


Рис. 2

Скорость передачи информации измеряется в количестве переданных за одну секунду бит или в бодах (baud): 1 бод = 1 бит/сек (bps). Производные единицы для бода такие же как и для бита и байта, например, 10 Kbaud = 10240 baud.

Информацию можно передавать *последовательно*, т. е. бит за битом, и *параллельно*, т. е. группами фиксированного количества бит. Параллельный способ быстрее, но он часто технически сложнее и дороже особенно при передаче данных на большие расстояния. Параллельный способ передачи используют, как правило, только на расстоянии не более 5 метров.

▶ Упражнение 3
Сколько бит в одном килобайте?

5. **Базовые понятия теории информации**

Информация — нематериальная сущность, при помощи которой с любой точностью можно описывать реальные (материальные), виртуальные (возможные) и понятийные сущности. Информация — противоположность неопределенности.

Канал связи — это среда передачи информации, которая характеризуется в первую очередь максимально возможной для нее скоростью передачи данных (*емкостью* канала связи).

Шум — это помехи в канале связи при передаче информации.

Кодирование — преобразование дискретной информации одним из следующих способов: шифрование, сжатие, защита от шума.

Общая схема передачи информации изображена на рис. 3.

Емкость канала связи без шума можно приблизительно вычислить, зная максимальную частоту волновых процессов, допустимую в этом канале. Можно считать, что скорость передачи данных может быть не меньше, чем эта частота. Например, при предельной частоте, равной 1000 Гц, можно обеспечить скорость передачи данных не меньше 1 Кбод.

Примеры каналов связи и связанных с ними предельных частот: телеграф — 140 Гц, телефон — до 3.1 КГц, короткие волны (10–100 м)

— 3–30 МГц, УКВ (1–10 м) — 30–300 МГц, спутник (сантиметровые волны) — до 30 ГГц, оптический (инфракрасный диапазон) — 0.15–400 ТГц, оптический (видимый свет) — 400–700 ТГц, оптический (ультрафиолетовый диапазон) — 0.7–1.75 ПГц.

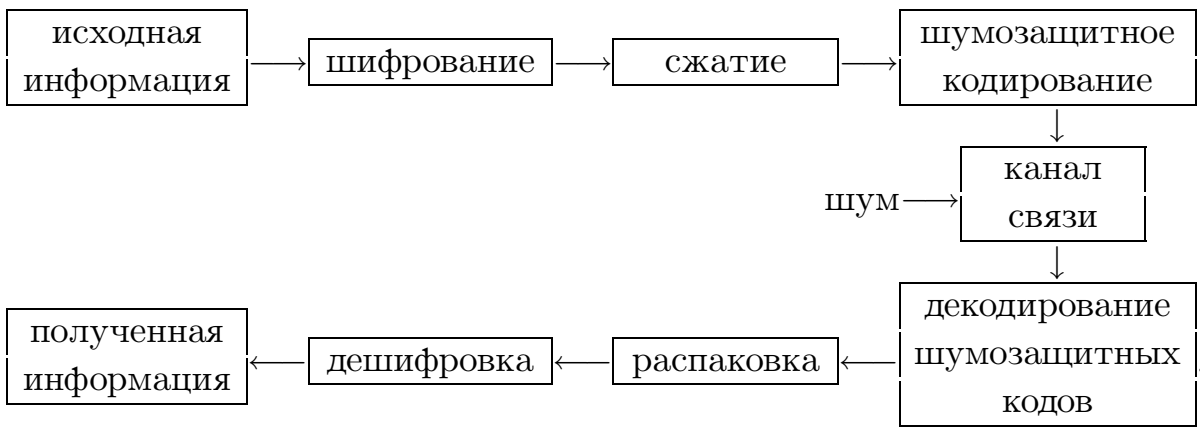


Рис. 3

Типичные современные каналы: телеграфный и телефонный. Перспективные, внедряемые ныне: оптоволоконный (терабоды) и цифровой телефонный (ISDN, Integrated Services Digital Networks) — 57–128 Кбод.

В реальных оптоволоконных системах скорость гораздо ниже теоретических пределов (редко превосходит 1–10 Гбод).

Наиболее широко пока используются телефонные линии связи. Здесь достигнута скорость более 50 Кбод!

6. **Способы измерения информации**

Понятие количества информации естественно возникает, например, в следующих типовых случаях:

1. Равенство вещественных переменных $a = b$, заключает в себе информацию о том, что a равно b. Про равенство $a^2 = b^2$ можно сказать, что оно несет меньшую информацию, чем первое, т. к. из первого следует второе, но не наоборот. Равенство $a^3 = b^3$ несет в себе информацию по объему такую же, как и первое;

2. Пусть происходят некоторые измерения с некоторой погрешностью. Тогда чем больше будет проведено измерений, тем больше информации об измеряемой сущности будет получено;

3. М. о. некоторой сл. в. содержит в себе информацию о самой сл. в. Для сл. в., распределенной по нормальному закону, с известной дисперсией знание м. о. дает полную информацию о сл. в.;

4. Рассмотрим схему передачи информации. Пусть передатчик описывается сл. в. X, тогда из-за помех в канале связи на приемник будет приходить сл. в. $Y = X + Z$, где Z — это сл. в., описывающая помехи. В этой схеме можно говорить о количестве информации, содержащейся в сл. в. Y, относительно X. Чем ниже уровень помех (дисперсия Z мала),

тем больше информации можно получить из Y. При отсутствии помех Y содержит в себе всю информацию об X.

В 1865 г. немецкий физик Рудольф Клаузиус ввел в статистическую физику понятие энтропии или меры уравновешенности системы.

В 1921 г. основатель большей части математической статистики, англичанин Роналд Фишер впервые ввел термин "информация" в математику, но полученные им формулы носят очень специальный характер.

В 1948 г. Клод Шеннон в своих работах по теории связи выписывает формулы для вычисления количества информация и энтропии. Термин "энтропия" используется Шенноном по совету патриарха компьютерной эры фон Неймана, отметившего, что полученные Шенноном для теории связи формулы для ее расчета совпали с соответствующими формулами статистической физики, а также то, что "точно никто не знает" что же такое энтропия.

▶ Упражнение 4
Какое из соотношений несет в себе больше информации $x = 5$ или $x > 3$?

7. Вероятностный подход к измерению дискретной и непрерывной информации

В основе теории информации лежит предложенный Шенноном способ измерения количества информации, содержащейся в одной сл. в. относительно другой сл. в. Этот способ приводит к выражению количества информации числом.

Для д. с. в. X и Y, заданных законами распределения $P(X = X_i) = p_i$, $P(Y = Y_j) = q_j$ и совместным распределением $P(X = X_i, Y = Y_j) = p_{ij}$, *количество информации*, содержащейся в X относительно Y, равно

$$I(X, Y) = \sum_{i,j} p_{ij} \log_2 \frac{p_{ij}}{p_i q_j}.$$

Для непрерывных сл. в. X и Y, заданных плотностями распределения вероятностей $p_X(t_1)$, $p_Y(t_2)$ и $p_{XY}(t_1, t_2)$, аналогичная формула имеет вид

$$I(X, Y) = \iint\limits_{\mathbb{R}^2} p_{XY}(t_1, t_2) \log_2 \frac{p_{XY}(t_1, t_2)}{p_X(t_1) p_Y(t_2)} dt_1 dt_2.$$

Очевидно, что

$$P(X = X_i, X = X_j) = \begin{cases} 0, & \text{при } i \neq j \\ P(X = X_i), & \text{при } i = j \end{cases}$$

и, следовательно,

$$I(X,X) = \sum_i p_i \log_2 \frac{p_i}{p_i p_i} = -\sum_i p_i \log_2 p_i.$$

Энтропия д.с.в. X в теории информации определяется формулой

$$H(X) = HX = I(X,X).$$

Свойства меры информации и энтропии:

1) $I(X,Y) \geqslant 0$, $I(X,Y) = 0 \quad \Leftrightarrow \quad X$ и Y независимы;
2) $I(X,Y) = I(Y,X)$;
3) $HX = 0 \quad \Leftrightarrow \quad X$ — константа;
4) $I(X,Y) = HX + HY - H(X,Y)$, где $H(X,Y) = -\sum_{i,j} p_{ij} \log_2 p_{ij}$;
5) $I(X,Y) \leqslant I(X,X)$. Если $I(X,Y) = I(X,X)$, то X — функция от Y.

1) Логарифмированием из очевидного для всех x неравенства $e^{x-1} \geqslant x$ (равенство устанавливается только при $x = 1$) получается неравенство $x - 1 \geqslant \ln x$ или $\frac{x-1}{\ln 2} \geqslant \log_2 x$.

$$-I(X,Y) = \sum_{i,j} p_{ij} \log_2 \frac{p_i q_j}{p_{ij}} \leqslant \sum_{i,j} p_{ij} \frac{\frac{p_i q_j}{p_{ij}} - 1}{\ln 2} =$$

$$= \sum_{i,j} \frac{p_i q_j - p_{ij}}{\ln 2} = \frac{\sum_i p_i \sum_j q_j - \sum_{i,j} p_{ij}}{\ln 2} = \frac{1-1}{\ln 2} = 0,$$

т.е. $I(X,Y) = 0$ только при $p_{ij} = p_i q_j$ для всех i и j, т.е. при независимости X и Y. Если X и Y независимы, то $p_{ij} = p_i q_j$ и, следовательно, аргументы логарифмов равны 1 и, следовательно, сами логарифмы равны 0, что означает, что $I(X,Y) = 0$;

2) Следует из симметричности формул относительно аргументов;

3) Если $HX = 0$, то все члены суммы, определяющей HX, должны быть нули, что возможно только тогда и только тогда, когда X — константа;

4) Из четырех очевидных соотношений

$$\sum_j p_{ij} = p_i, \quad \sum_i p_{ij} = q_j,$$

$$HX = -\sum_i p_i \log_2 p_i = -\sum_{i,j} p_{ij} \log_2 p_i,$$

$$HY = -\sum_j q_j \log_2 q_j = -\sum_{i,j} p_{ij} \log_2 q_j$$

получается

$$HX + HY - H(X,Y) = \sum_{i,j} p_{ij}(\log_2 p_{ij} - \log_2 q_j - \log_2 p_i) = I(X,Y);$$

5) Нужно доказать $I(X,Y) = HX + HY - H(X,Y) \leqslant HX$ или $HY - H(X,Y) \leqslant 0$.

$$HY - H(X,Y) = -\sum_{i,j} p_{ij} \log_2 q_j + \sum_{i,j} p_{ij} \log_2 p_{ij} = \sum_{i,j} p_{ij} \log_2(p_{ij}/q_j),$$

но $p_{ij} = P(X = X_i, Y = Y_j) \leqslant q_j = P(Y = Y_j)$, а значит аргументы у всех логарифмов не больше 1 и, следовательно, значения логарифмов не больше 0, а это и значит, что вся сумма не больше 0.

Если $HX = I(X,X) = I(X,Y)$, то для каждого i p_{ij} равно либо q_j, либо 0. Но из $p_{ij} = P(X = X_i, Y = Y_j) = P(X = X_i/Y = Y_j)P(Y = Y_j) \in \{q_j, 0\}$ следует $P(X = X_i/Y = Y_j) \in \{0, 1\}$, что возможно только в случае, когда X — функция от Y.

При независимости сл. в. X и Y одна из них ничем не описывает другую, что и отражается в том, что для таких сл.в. $I(X,Y) = 0$.

Рассмотрим пример измерения количества информации при подбрасывании двух игральных костей.

Пусть заданы д. с. в. X_1, X_2 и Y. X_1 и X_2 — количества очков, выпавших соответственно на 1-й и 2-й игральной кости, а $Y = X_1 + X_2$. Найти $I(Y, X_1)$, $I(X_1, X_1)$, $I(Y,Y)$.

Законы распределения вероятностей для д. с. в. X_1 и X_2 совпадают, т. к. кости одинаковые и без изъянов.

X_1	1	2	3	4	5	6
p	1/6					

, т. е. при $j = 1...6$ $q_j = P(X_1 = j) = 1/6$.

Закон распределения вероятностей для д. с. в. Y,

$$P(Y = i) = P(X_1 + X_2 = i), \quad i = 2...12,$$

вследствие того, что X_1, X_2 — независимы и поэтому

$$P(X_1 = n, X_2 = m) = P(X_1 = n)P(X_2 = m),$$

будет

$$p_i = P(X_1 + X_2 = i) = \sum_{\substack{n+m=i \\ 1 \leqslant n,m \leqslant 6}} P(X_1 = n)P(X_2 = m) = \sum_{\substack{n+m=i \\ 1 \leqslant n,m \leqslant 6}} 1/36.$$

Таблицы, определяющие Y:

$X_2 \backslash X_1$	1	2	3	4	5	6
1	2	3	4	5	6	7
2	3	4	5	6	7	8
3	4	5	6	7	8	9
4	5	6	7	8	9	10
5	6	7	8	9	10	11
6	7	8	9	10	11	12,

$Y = X_1 + X_2$	2	3	4	5	6	7	8	9	10	11	12
p	$1/36$	$2/36$	$3/36$	$4/36$	$5/36$	$6/36$	$5/36$	$4/36$	$3/36$	$2/36$	$1/36$,

т.е. при $i = 2...12$, $p_i = P(Y = i) = (6 - |7 - i|)/36$.

Закон совместного распределения вероятностей д.с.в. X_1 и Y будет

$$p_{ij} = P(Y = i, X_1 = j) = P(Y = i/X_1 = j)P(X_1 = j),$$

например,

$$P(Y = 2, X_1 = 1) = P(Y = 2/X_1 = 1)P(X_1 = 1) =$$

$$= P(X_2 = 1)P(X_1 = 1) = 1/36.$$

В общем случае получится

$$p_{ij} = P(Y = i, X_1 = j) = \begin{cases} 1/36, & \text{при } 1 \leqslant i - j \leqslant 6, \\ 0, & \text{иначе.} \end{cases}$$

$X_1 \backslash Y$	2	3	4	5	6	7	8	9	10	11	12
1	$1/36$	$1/36$	$1/36$	$1/36$	$1/36$	$1/36$	0	0	0	0	0
2	0	$1/36$	$1/36$	$1/36$	$1/36$	$1/36$	$1/36$	0	0	0	0
3	0	0	$1/36$	$1/36$	$1/36$	$1/36$	$1/36$	$1/36$	0	0	0
4	0	0	0	$1/36$	$1/36$	$1/36$	$1/36$	$1/36$	$1/36$	0	0
5	0	0	0	0	$1/36$	$1/36$	$1/36$	$1/36$	$1/36$	$1/36$	0
6	0	0	0	0	0	$1/36$	$1/36$	$1/36$	$1/36$	$1/36$	$1/36$

Тогда

$$I(Y, X_1) = \sum_{j=1}^{6} \sum_{1 \leqslant i-j \leqslant 6} p_{ij} \log_2 \frac{p_{ij}}{p_i q_j} =$$

$$= \frac{1}{36} \sum_{j=1}^{6} \sum_{1 \leqslant i-j \leqslant 6} \log_2 \frac{1}{6p_i} =$$

$$= \frac{1}{36}\left(\sum_{i=2}^{7} \log_2 \frac{1}{6p_i} + \sum_{i=3}^{8} \log_2 \frac{1}{6p_i} + \cdots + \sum_{i=6}^{11} \log_2 \frac{1}{6p_i} + \sum_{i=7}^{12} \log_2 \frac{1}{6p_i}\right) =$$

$$= \frac{1}{36}((\log_2 \frac{6}{1}+\log_2 \frac{6}{2}+\cdots+\log_2 \frac{6}{6})+\cdots+(\log_2 \frac{6}{6}+\log_2 \frac{6}{5}+\cdots+\log_2 \frac{6}{1})) =$$

$$= \frac{1}{36}(\underline{2\log_2 6} + 4\log_2 3 + 6\log_2 2 + 8\log_2 \frac{3}{2} + 10\log_2 \frac{6}{5} + \underline{6\log_2 1}) =$$

$$= (2 + 2\log_2 3 + 4\log_2 3 + 6 + 8\log_2 3 - 8 + 10\log_2 3 + 10 - 10\log_2 5)/36 =$$

$$= (10 + 24\log_2 3 - 10\log_2 5)/36 \approx 0.69 \text{ бит/символ.}$$

$I(X_1, X_1) = I(X_2, X_2) = -\sum_{j=1}^{6} q_j \log_2 q_j = \log_2 6 = 1 + \log_2 3 \approx 2.58$ бит/сим.

$I(Y, Y) = -\sum_{i=2}^{12} p_i \log_2 p_i =$

$$= \frac{1}{36}(2\log_2 36 + 4\log_2 18 + 6\log_2 12 + 8\log_2 9 + 10\log_2 \frac{36}{5} + 6\log_2 6) =$$

$$= (4+4\log_2 3+4+8\log_2 3+12+6\log_2 3+16\log_2 3+20+20\log_2 3-10\log_2 5+$$

$$+ 6 + 6\log_2 3)/36 = (46 + 60\log_2 3 - 10\log_2 5)/36 \approx 3.27 \text{ бит/сим.}$$

Здесь $0 < I(Y, X_1) = I(Y, X_2) < I(X_1, X_1) = I(X_2, X_2) < I(Y, Y)$, что соответствует свойствам информации.

Подчекнутый член $\frac{1}{36}2\log_2 6 = I(X_1, X_1)/18$ в расчете $I(X_1, Y)$ соответствует информации о двух случаях из 36, когда $Y = 2$ и $Y = 12$, которые однозначно определяют X_1. Шесть случаев, когда $Y = 7$, не несут никакой информации об X_1, что соответствует подчеркнутому члену $6\log_2 1 = 0$.

Расчеты можно проводить, используя 4-е свойство информации, через энтропию.

$H(Y, X_1) = -\sum_{i,j} p_{ij} \log_2 p_{ij} = \log_2 36 = 2(1 + \log_2 3) = 2HX_1 \approx 5.17$ бит/сим.

$I(Y, X_1) = HX_1 + HY - H(X_1, Y) = HY - HX_1 \approx 3.27 - 2.58 = 0.69$ бит/сим.

Расчет количества информации с использованием 4-го свойства, а не определения, обычно требует меньше вычислений.

Рассмотрим более простой пример. Пусть д. с. в. X равна количеству очков, выпавших на игральной кости, а д. с. в. Y равна 0, если выпавшее количество очков нечетно, и 1, если выпавшее количество очков четно. Найти $I(X, Y)$ и $I(Y, Y)$.

Составим законы распределения вероятностей д. с. в. X и Y.

X	1	2	3	4	5	6
p	1/6					

Y	0	1
p	1/2	

Таким образом, при $i = 1...6$ $p_i = P(X = i) = 1/6$ и, соответственно, при $j = 0...1$ $q_j = P(Y = j) = 1/2$.

Составим также закон совместного распределения вероятностей этих д. с. в.

X	1	3	5	2	4	6	1	3	5	2	4	6
Y	0	0	0	1	1	1	1	1	1	0	0	0
p	1/6						0					

Таким образом, $p_{ij} = P(X = i, Y = j) = \begin{cases} 0, & \text{если } i+j \text{ — четно,} \\ 1/6, & \text{иначе.} \end{cases}$

$I(X,Y) = \sum_{i,j} p_{ij} \log_2 \frac{p_{ij}}{p_i q_j} = 6\frac{1}{6} \log_2 2 = 1$ бит/сим.

$I(Y,Y) = -\sum_{j=0}^{1} q_j \log_2 q_j = 2\frac{1}{2} \log_2 2 = 1$ бит/сим.

Точное количество выпавших очков дает точную информацию о четности, т. е. 1 бит. Из $I(X,Y) = I(Y,Y) = 1$ бит/сим и 3-го свойства информации следует, что информация об X полностью определяет Y, но не наоборот, т. к. $I(X,Y) \neq I(X,X) = 1 + \log_2 3 \approx 2.58$ бит/сим. Действительно, Y функционально зависит от X, а X от Y функционально не зависит.

Расчеты через энтропию будут следующими

$H(X,Y) = -\sum_{i,j} p_{ij} \log_2 p_{ij} = \log_2 6 = 1 + \log_2 3 = HX$,

$I(X,Y) = HX + HY - HX = HY = 1$ бит/сим.

▶ Упражнение 5

Найти энтропию д. с. в. X, заданной распределением

X	1	2	3	4	5	6	7	8
p	0.1	0.2	0.1	0.05	0.1	0.05	0.3	0.1.

▶ Упражнение 6

Значения д. с. в. X_1 и X_2 определяются подбрасыванием двух идеальных монет, а д. с. в. Y равна сумме количества "гербов", выпавших при подбрасывании этих монет. Сколько информации об X_1 содержится в Y?

▶ Упражнение 7

Сколько информации об X_1 содержится в д. с. в. $Z = (X_1 + 1)^2 - X_2$, где независимые д. с. в. X_1 и X_2 могут с равной вероятностью принимать значение либо 0, либо 1? Найти HX_1 и HZ. Каков характер зависимости между X_1 и Z?

▶ Упражнение 8

Д. с. в. X_1, X_2 — зависимы и распределены также как и соответствующие д. с. в. из предыдущей задачи. Найти $I(X_1, X_2)$, если совместное распределение вероятностей X_1 и X_2 описывается законом

X_1	0	0	1	1
X_2	0	1	0	1
p	1/3	1/6	1/6	1/3.

▶ Упражнение 9
Д. с. в. X_1 и X_2 определяются подбрасыванием двух идеальных тетраэдров, грани которых помечены числами от 1 до 4. Д. с. в. Y равна сумме чисел, выпавших при подбрасывании этих тетраэдров, т. е. $Y = X_1 + X_2$. Вычислить $I(X_1, Y)$, HX_1 и HY.

▶ Упражнение 10
Подсчитать сколько информации об X_1 содержится в д. с. в. $Z = X_1 * X_2$, а также HZ. Д. с. в. X_1 и X_2 берутся из предыдущего упражнения.

▶ Упражнение 11
Д. с. в. X_1 может принимать три значения -1, 0 и 1 с равными вероятностями. Д. с. в. X_2 с равными вероятностями может принимать значения 0, 1 и 2. X_1 и X_2 — независимы. $Y = X_1^2 + X_2$. Найти $I(X_1, Y)$, $I(X_2, Y)$, HX_1, HX_2, HY.

▶ Упражнение 12
Найти энтропии д. с. в. X, Y, Z и количество информации, содержащейся в $Z = X + Y$ относительно Y. X и Y — независимы и задаются распределениями

X	0	1	3	4
p	1/8	1/8	1/4	1/2

Y	-2	2
p	3/8	5/8.

8. Смысл энтропии Шеннона

Энтропия д. с. в. — это минимум среднего количества бит, которое нужно передавать по каналу связи о текущем значении данной д. с. в.

Рассмотрим пример (скачки). В заезде участвуют 4 лошади с равными шансами на победу, т. е. вероятность победы каждой лошади равна 1/4. Введем д. с. в. X, равную номеру победившей лошади. Здесь $HX = 2$. После каждого заезда по каналам связи достаточно будет передавать два бита информации о номере победившей лошади. Кодируем номер лошади следующим образом: 1—00, 2—01, 3—10, 4—11. Если ввести функцию $L(X)$, которая возвращает длину сообщения, кодирующего заданное значение X, то м. о. $ML(X)$ — это средняя длина сообщения, кодирующего X. Можно формально определить L через две функции $L(X) = len(code(X))$, где $code(X)$ каждому значению X ставит в соответствие некоторый битовый код, причем, взаимно однозначно, а len возвращает длину в битах для любого конкретного кода. L — это функция от д. с. в., т. е. тоже д. с. в. В этом примере $ML(X) = HX$.

Пусть теперь д. с. в. X имеет следующее распределение

$$P(X = 1) = \frac{3}{4},\ P(X = 2) = \frac{1}{8},\ P(X = 3) = P(X = 4) = \frac{1}{16},$$

т.е. лошадь с номером 1 — это фаворит. Тогда

$$HX = \frac{3}{4}\log_2\frac{4}{3} + \frac{1}{8}\log_2 8 + \frac{1}{8}\log_2 16 = \frac{19}{8} - \frac{3}{4}\log_2 3 \approx 1.186 \text{ бит/сим.}$$

Закодируем номера лошадей: 1—0, 2—10, 3—110, 4—111, — т.е. так, чтобы каждой код не был префиксом другого кода (подобное кодирование называют *префиксным*). В среднем в 16 заездах 1-я лошадь должна победить в 12 из них, 2-я — в 2-х, 3-я — в 1-м и 4-я — в 1-м. Таким образом, средняя длина сообщения о победителе равна $(1 * 12 + 2 * 2 + 3 * 1 + 3 * 1)/16 = 1.375$ бит/сим или м.о. $L(X)$. Действительно, $L(X)$ сейчас задается следующим распределением вероятностей: $P(L(X) = 1) = 3/4$, $P(L(X) = 2) = 1/8$, $P(L(X) = 3) = 1/8$. Следовательно,

$$ML(X) = \frac{3}{4} + \frac{2}{8} + \frac{3}{8} = \frac{11}{8} = 1.375 \text{ бит/сим.}$$

Итак, $ML(X) > HX$.

Можно доказать, что более эффективного кодирования для двух рассмотренных случаев не существует.

То, что энтропия Шеннона соответствует интуитивному представлению о мере информации, может быть продемонстрировано в опыте по определению среднего времени психических реакций. Опыт заключается в том, что перед испытуемым человеком зажигается одна из N лампочек, которую он должен указать. Проводится большая серия испытаний, в которых каждая лампочка зажигается с определенной вероятностью p_i ($\sum_i^N p_i = 1$), где i — это номер лампочки. Оказывается, среднее время, необходимое для правильного ответа испытуемого, пропорционально величине энтропии $-\sum_{i=1}^N p_i \log_2 p_i$, а не числу лампочек N, как можно было бы подумать. В этом опыте предполагается, что чем больше информации будет получено человеком, тем дольше будет время ее обработки и, соответственно, реакции на нее.

► Упражнение 13

Найти энтропию д.с.в. X и среднюю длину каждого из приведенных кодов для этой д.с.в.

X	1	3	4	5	6
p	0.4	0.2	0.1	0.2	0.1
$code1(X)$	000	001	010	011	111
$code2(X)$	0	100	101	110	111
$code3(X)$	00	01	110	10	111
$code4(X)$	0	10	1110	110	1111.

▶ Упражнение 14
Д. с. в. X равна количеству "гербов", выпавших на двух идеальных монетках. Найти энтропию X. Придумать минимальный код для X, вычислить его среднюю длину и обосновать его минимальность.

▶ Упражнение 15
Д. с. в. X задана распределением $P(X = 2^n) = 1/2^n$, $n = 1, 2, \ldots$ Найти энтропию этой д. с. в. Придумать минимальный код для X, вычислить его среднюю длину и обосновать его минимальность.

▶ Упражнение 16
Про д. с. в. X известно, что ее значениями являются буквы кириллицы. Произведен ряд последовательных измерений X, результат которых — "ТЕОРИЯИНФОРМАЦИИ". Составить на основании этого результата приблизительный закон распределения вероятностей этой д. с. в. и оценить минимальную среднюю длину кодов для X.

9. **Семантическая информация**

В 50-х годах XX века появились первые попытки определения абсолютного информационного содержания предложений естественного языка. Стоит отметить, что сам Шеннон однажды заметил, что смысл сообщений не имеет никакого отношения к его теории информации, целиком построенной на положениях теории вероятностей. Но его способ точного измерения информации наводил на мысль о возможности существования способов точного измерения информации более общего вида, например, информации из предложений естественного языка. Примером одной из таких мер является функция $inf(s) = -\log_2 p(s)$, где s — это предложение, смысловое содержание которого измеряется, $p(s)$ — вероятность истинности s. Вот некоторые свойства этой функции-меры:

1) если $s_1 \Rightarrow s_2$ (из s_1 следует s_2) — истинно, то $inf(s_1) \geqslant inf(s_2)$;
2) $inf(s) \geqslant 0$;
3) если s — истинно, то $inf(s) = 0$;
4) $inf(s_1 s_2) = inf(s_1) + inf(s_2) \quad \Leftrightarrow \quad p(s_1 \cdot s_2) = p(s_1)p(s_2)$, т. е. независимости s_1 и s_2.

Значение этой функция-меры больше для предложений, исключающих большее количество возможностей. Пример: из s_1 — "$a > 3$" и s_2 — "$a = 7$" следует, что $s_2 \Rightarrow s_1$ или $inf(s_2) \geqslant inf(s_1)$; ясно, что s_2 исключает больше возможностей, чем s_1.

Для измерения семантической информации также используется функция-мера $cont(s) = 1 - p(s)$. Ясно, что $cont(s) = 1 - 2^{-inf(s)}$ или $inf(s) = -\log_2(1 - cont(s))$.

► **Упражнение 17**
Вычислить $inf(s)$ и $cont(s)$ предложения s_1, про которое известно, что оно достоверно на 50%, и предложения s_2, достоверность которого 25%.

10. **Сжатие информации**

Цель сжатия — уменьшение количества бит, необходимых для хранения или передачи заданной информации, что дает возможность передавать сообщения более быстро и хранить более экономно и оперативно (последнее означает, что операция извлечения данной информации с устройства ее хранения будет проходить быстрее, что возможно, если скорость распаковки данных выше скорости считывания данных с носителя информации). Сжатие позволяет, например, записать больше информации на дискету, "увеличить" размер жесткого диска, ускорить работу с модемом и т. д. При работе с компьютерами широко используются программы-архиваторы данных формата ZIP, GZ, ARJ и других. Методы сжатия информации были разработаны как математическая теория, которая долгое время (до первой половины 80-х годов), мало использовалась в компьютерах на практике.

Сжатие данных не может быть большим некоторого теоретические предела. Для формального определения этого предела рассматриваем любое информационное сообщение длины n как последовательность независимых, одинаково распределенных д. с. в. X_i или как выборки длины n значений одной д. с. в. X.

Доказано [20], что среднее количество бит, приходящихся на одно кодируемое значение д. с. в., не может быть меньшим, чем энтропия этой д. с. в., т. е. $ML(X) \geqslant HX$ для любой д. с. в. X и любого ее кода.

Кроме того, доказано [20] утверждение о том, что существует такое кодирование (Шеннона-Фэно, Fano), что $HX \geqslant ML(X) - 1$.

Рассмотрим д. с. в. X_1 и X_2, независимые и одинаково распределенные. $HX_1 = HX_2$ и $I(X_1, X_2) = 0$, следовательно,

$$H(X_1, X_2) = HX_1 + HX_2 - I(X_1, X_2) = 2HX_1.$$

Вместо X_1 и X_2 можно говорить о двумерной д. с. в. $\vec{X} = (X_1, X_2)$. Аналогичным образом для n-мерной д. с. в. $\vec{X} = (X_1, X_2, \ldots, X_n)$ можно получить, что $H\vec{X} = nHX_1$.

Пусть $L_1(\vec{X}) = L(\vec{X})/n$, где $\vec{X} = (X_1, X_2, \ldots, X_n)$, т. е. $L_1(\vec{X})$ — это количество бит кода на единицу сообщения $\vec{X}$. Тогда $ML_1(\vec{X})$ — это среднее количество бит кода на единицу сообщения при передаче бесконечного множества сообщений $\vec{X}$. Из $ML(\vec{X}) - 1 \leqslant H\vec{X} \leqslant ML(\vec{X})$ для кода Шеннона-Фэно для $\vec{X}$ следует $ML_1(\vec{X}) - 1/n \leqslant HX_1 \leqslant ML_1(\vec{X})$ для этого же кода.

Таким образом, доказана *основная теорема о кодировании при отсутствии помех*, а именно то, что с ростом длины n сообщения, при кодировании методом Шеннона-Фэно всего сообщения целиком среднее количество бит на единицу сообщения будет сколь угодно мало отличаться от энтропии единицы сообщения. Подобное кодирование практически не реализуемо из-за того, что с ростом длины сообщения трудоемкость построения этого кода становится недопустимо большой. Кроме того, такое кодирование делает невозможным отправку сообщения по частям, что необходимо для непрерывных процессов передачи данных. Дополнительным недостатком этого способа кодирования является необходимость отправки или хранения собственно полученного кода вместе с его исходной длиной, что снижает эффект от сжатия. На практике для повышения степени сжатия используют *метод блокирования*.

По выбранному значению $\varepsilon > 0$ можно выбрать такое s, что если разбить все сообщение на блоки длиной s (всего будет n/s блоков), то кодированием Шеннона-Фэно таких блоков, рассматриваемых как единицы сообщения, можно сделать среднее количество бит на единицу сообщения бóльшим энтропии менее, чем на ε. Действительно, пусть $\vec{Y} = (\vec{Y}_1, \vec{Y}_2, \ldots, \vec{Y}_{n/s})$, $\vec{Y}_1 = (X_1, X_2, \ldots, X_s)$, $\vec{Y}_2 = (X_{s+1}, X_{s+2}, \ldots, X_{2s})$ и т. д., т. е. $\vec{Y}_i = (X_{s(i-1)+1}, X_{s(i-1)+2}, \ldots, X_{si})$. Тогда $H\vec{Y}_1 = sHX_1$ и $sML_1(\vec{Y}_1) = ML(\vec{Y}_1) \leqslant H\vec{Y}_1 + 1 = sHX_1 + 1$, следовательно,

$$ML_1(\vec{Y}_1) \leqslant HX_1 + 1/s,$$

т. е. достаточно брать $s = 1/\varepsilon$. Минимум s по заданному ε может быть гораздо меньшим $1/\varepsilon$.

Пример. Пусть д. с. в. $X_1, X_2, \ldots X_n$ независимы и одинаково распределены и могут принимать только два значения $P(X_i = 0) = p = 3/4$ и $P(X_i = 1) = q = 1/4$ при i от 1 до n. Тогда

$$HX_i = \frac{3}{4}\log_2\frac{4}{3} + \frac{1}{4}\log_2 4 = 2 - \frac{3}{4}\log_2 3 \approx 0.811 \text{ бит/сим.}$$

Минимальное кодирование здесь — это коды 0 и 1 с длиной 1 бит каждый. При таком кодировании количество бит в среднем на единицу сообщения равно 1. Разобьем сообщение на блоки длины 2. Закон распределения вероятностей и кодирование для 2-мерной д. с. в. $\vec{X} = (X_1, X_2)$ —

$\vec{X}$	00	01	10	11
p	9/16	3/16	3/16	1/16
$code(\vec{X})$	0	10	110	111
$L(\vec{X})$	1	2	3	3.

Тогда при таком минимальном кодировании количество бит в среднем на единицу сообщения будет уже

$$ML_1(\vec{X}) = \big(1\frac{9}{16} + 2\frac{3}{16} + 3\frac{3}{16} + 3\frac{1}{16}\big)/2 = \frac{27}{32} = 0.84375,$$

т. е. меньше, чем для неблочного кодирования. Для блоков длины 3 количество бит в среднем на единицу сообщения можно сделать ≈ 0.823, для блоков длины 4 — ≈ 0.818 и т. д.

Все изложенное ранее подразумевало, что рассматриваемые д. с. в. кодируются только двумя значениями (обычно 0 и 1). Пусть д. с. в. кодируются m значениями. Тогда для д. с. в. $\vec{X}$ и любого ее кодирования верно, что $ML(\vec{X}) \geqslant H\vec{X}/\log_2 m$ и $ML_1(\vec{X}) \geqslant HX_1/\log_2 m$. Кроме того, существует кодирование такое, что $ML(\vec{X}) - 1 \leqslant H\vec{X}/\log_2 m$ и $ML_1(\vec{X}) - 1/n \leqslant HX_1/\log_2 m$, где $n = \dim(\vec{X})$.

Формулы теоретических приделов уровня сжатия, рассмотренные ранее, задают предел для средней длины кода на единицу сообщений, передаваемых много раз, т. е. они ничего не говорят о нижней границе уровня сжатия, которая может достигаться на некоторых сообщениях и быть меньшей энтропии д. с. в., реализующей сообщение.

11. **Простейшие алгоритмы сжатия информации**

Метод Шеннона-Фэно состоит в следующем, значения д. с. в. располагают в порядке убывания их вероятностей, а затем последовательно делят на две части с приблизительно равными вероятностями, к коду первой части добавляют 0, а к коду второй — 1.

Для предшествующего примера получим

$\vec{X}$	p	$code(\vec{X})$
00	9/16	0
01	3/16	10
10	3/16	110
11	1/16	111,

$ML_1(\vec{X}) = 27/32 = 0.84375$ бит/сим.

Еще один пример. Код составляется после сортировки, т. е. после перестановки значений B и C.

X	p	$code(X)$
A	0.4	0
B	0.2	11
C	0.4	10,

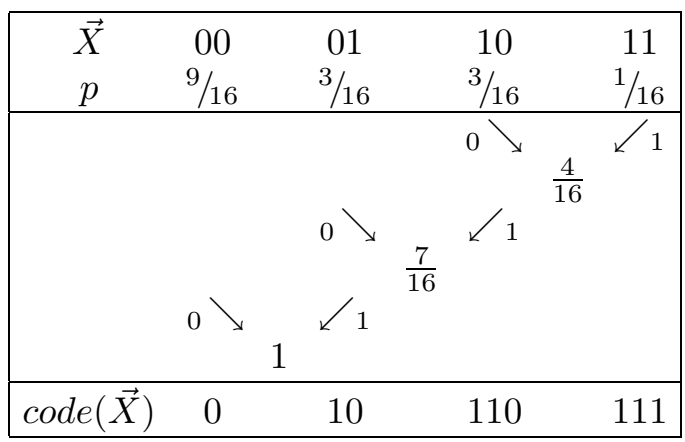
$ML(X) = ML_1(X) = 1.6$ бит/сим,
$HX = \log_2 5 - 0.8 \approx 1.523$ бит/сим.

Метод Хаффмена (Huffman) разработан в 1952 г. Он более практичен и никогда по степени сжатия не уступает методу Шеннона-Фэно, более того, он сжимает максимально плотно. Код строится при помощи

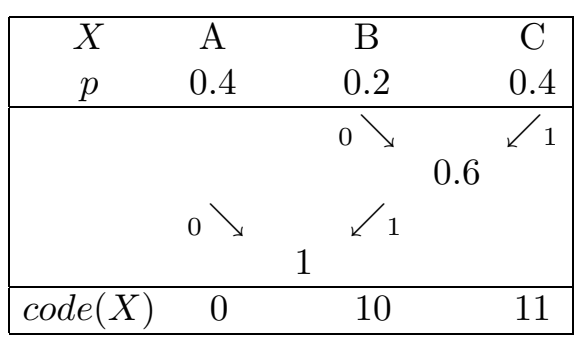
двоичного (бинарного) дерева. Вероятности значений д. с. в. приписываются его листьям; все дерево строится, опираясь на листья. Величина, приписанная к узлу дерева, называется весом узла. Два листа с наименьшими весами создают родительский узел с весом, равным сумме их весов; в дальнейшем этот узел учитывается наравне с оставшимися листьями, а образовавшие его узлы от такого рассмотрения устраняются. После постройки корня нужно приписать каждой из ветвей, исходящих из родительских узлов, значения 0 или 1. Код каждого значения д. с. в. — это число, получаемое при обходе ветвей от корня к листу, соответствующему данному значению.

Для методов Хаффмена и Шеннона-Фэно каждый раз вместе с собственно сообщением нужно передавать и таблицу кодов. Например, для случая из примера 2 нужно сообщить, что коду 10 соответствует символ C, коду 0 — A и т. д.

Построим коды Хаффмена для значений д. с. в. из двух предыдущих примеров.



$ML_1(\vec{X}) = ML(\vec{X})/2 = 27/32 = 0.84375$ бит/сим.



$ML_1(X) = ML(X) = 1.6$ бит/сим.

▶ Упражнение 18
Вычислить $ML_1(\vec{X})$ для блочного кода Хаффмена для X. Длина блока — 2 бита. Д. с. в. X берется из последнего примера.

▶ Упражнение 19
Вычислить HX и $ML(X)$ для кодов Хаффмена и Шеннона-Фэно для X.

Д. с. в. X задается следующим распределением вероятностей:

X	1	2	3	4	5
p	$^7/_{18}$	$^1/_6$	$^1/_6$	$^1/_6$	$^1/_9$.

12. **Арифметическое кодирование**

Алгоритм кодирования Хаффмена, в лучшем случае, не может передавать на каждый символ сообщения менее одного бита информации. Предположим, известно, что в сообщении, состоящем из нулей и единиц, единицы встречаются в 10 раз чаще нулей. При кодировании методом Хаффмена и на 0 и на 1 придется тратить не менее одного бита. Но энтропия д. с. в., генерирующей такие сообщения ≈ 0.469 бит/сим. Метод Хаффмена дает для минимального среднего количества бит на один символ сообщения значение 1 бит. Хотелось бы иметь такую схему кодирования, которая позволяла бы кодировать некоторые символы менее чем одним битом. Одной из лучших среди таких схем является арифметическое кодирование, разработанное в 70-х годах XX века.

По исходному распределению вероятностей для выбранной для кодирования д. с. в. строится таблица, состоящая из пересекающихся только в граничных точках отрезков для каждого из значений этой д. с. в.; объединение этих отрезков должно образовывать отрезок [0,1], а их длины должны быть пропорциональны вероятностям соответствующих значений д. с. в. Алгоритм кодирования заключается в построении отрезка, однозначно определяющего данную последовательность значений д. с. в. Затем для построенного отрезка находится число, принадлежащее его внутренней части и равное целому числу, деленному на минимально возможную положительную целую степень двойки. Это число и будет кодом для рассматриваемой последовательности. Все возможные конкретные коды — это числа строго большие нуля и строго меньшие одного, поэтому можно отбрасывать лидирующий ноль и десятичную точку, но нужен еще один специальный код-маркер, сигнализирующий о конце сообщения. Отрезки строятся так. Если имеется отрезок для сообщения длины $n-1$, то для построения отрезка для сообщения длины n, разбиваем его на столько же частей, сколько значений имеет рассматриваемая д. с. в. Это разбиение делается совершенно также как и самое первое (с сохранением порядка). Затем выбирается из полученных отрезков тот, который соответствует заданной конкретной последовательности длины n.

Принципиальное отличие этого кодирования от рассмотренных ранее методов в его непрерывности, т. е. в ненужности блокирования. Эффективность арифметического кодирования растет с ростом длины

сжимаемого сообщения (для кодирования Хаффмена или Шеннона-Фэно этого не происходит). Хотя арифметическое кодирование дает обычно лучшее сжатие, чем кодирование Хаффмена, оно пока используется на практике сравнительно редко, т.к. оно появилось гораздо позже и требует больших вычислительных ресурсов.

При сжатии заданных данных, например, из файла все рассмотренные методы требуют двух проходов. Первый для сбора частот символов, используемых как приближенные значения вероятностей символов, и второй для собственно сжатия.

Пример арифметического кодирования. Пусть д.с.в. X может принимать только два значения 0 и 1 с вероятностями 2/3 и 1/3 соответственно. Сопоставим значению 0 отрезок [0,2/3], а 1 — [2/3,1]. Тогда для д.с.в. $\vec{X}$,

$$\dim(\vec{X}) = 3,\ HX = H\vec{X}/3 = \log_2 3 - 2/3 \approx 0.9183 \text{ бит/сим.},$$

таблица построения кодов —

Интервалы и коды			Вероятность	Код Хаффмена
		$111[\frac{26}{27}, 1] \ni \frac{31}{32} = 0.11111$	$^1\!/_{27}$	0000
	$11[\frac{8}{9}, 1]$	$110[\frac{8}{9}, \frac{26}{27}] \ni \frac{15}{16} = 0.1111$	$^2\!/_{27}$	0001
		$101[\frac{22}{27}, \frac{8}{9}] \ni \frac{7}{8} = 0.111$	$^2\!/_{27}$	010
$1[\frac{2}{3}, 1]$	$10[\frac{2}{3}, \frac{8}{9}]$	$100[\frac{2}{3}, \frac{22}{27}] \ni \frac{3}{4} = 0.11$	$^4\!/_{27}$	001
		$011[\frac{16}{27}, \frac{2}{3}] \ni \frac{5}{8} = 0.101$	$^2\!/_{27}$	011
	$01[\frac{4}{9}, \frac{2}{3}]$	$010[\frac{4}{9}, \frac{16}{27}] \ni \frac{1}{2} = 0.1$	$^4\!/_{27}$	100
		$001[\frac{8}{27}, \frac{4}{9}] \ni \frac{3}{8} = 0.011$	$^4\!/_{27}$	101
$0[0, \frac{2}{3}]$	$00[0, \frac{4}{9}]$	$000[0, \frac{8}{27}] \ni \frac{1}{4} = 0.01$	$^8\!/_{27}$	11.

$$ML_1(\vec{X}) = 65/81 \approx 0.8025 \text{ бит/сим. (арифметическое)},$$

$$ML_1(\vec{X}) = 76/81 \approx 0.9383 \text{ бит/сим. (блочный Хаффмена)},$$

$$ML_1(X) = ML(X) = 1 \text{ бит/сим. (Хаффмена)}.$$

Среднее количество бит на единицу сообщения для арифметического кодирования получилось меньше, чем энтропия. Это связано с тем, что в рассмотренной простейшей схеме кодирования, не описан код-маркер конца сообщения, введение которого неминуемо сделает это среднее количество бит большим энтропии.

Получение исходного сообщения из его арифметического кода происходит по следующему алгоритму.

Шаг 1. В таблице для кодирования значений д. с. в. определяется интервал, содержащий текущий код, — по этому интервалу однозначно определяется один символ исходного сообщения. Если этот символ — это маркер конца сообщения, то конец.

Шаг 2. Из текущего кода вычитается нижняя граница содержащего его интервала, полученная разность делится на длину этого же интервала. Полученное число считается новым текущим значением кода. Переход к шагу 1.

▶ Упражнение 20

Вычислить среднее количество бит на единицу сжатого сообщения о значении каждой из д. с. в., из заданных следующими распределениями вероятностей, при сжатии методами Шеннона-Фэно, Хаффмена и арифметическим.

X_1	1	2	3	4
p	1/3	1/3	1/6	1/6,

X_2	1	2	5	6	7
p	0.2	0.1	0.3	0.25	0.15,

X_3	1	4	9	16	25	36	49
p	0.1	0.1	0.1	0.3	0.1	0.1	0.2,

X_4	-2	-1	0	1	2
p	1/3	1/4	1/5	1/6	1/20.

▶ Упражнение 21

Вычислить длины кодов Хаффмена и арифметического для сообщения AAB, полученного от д. с. в. X со следующим распределением вероятностей $P(X = A) = 1/3$, $P(X = B) = 2/3$.

▶ Упражнение 22

Составить арифметический код для сообщения BAABC, полученного от д. с. в. X со следующим распределением вероятностей $P(X = A) = 1/4$, $P(X = B) = 1/2$, $P(X = C) = 1/4$. Каков будет арифметический код для этого же сообщения, если X распределена по закону $P(X = A) = 1/3$, $P(X = B) = 7/15$, $P(X = C) = 1/5$?

▶ Упражнение 23

Д. с. в. X может принимать три различных значения. При построении блочного кода с длиной блока 4 для X необходимо будет рассмотреть д. с. в. $\vec{X}$ — выборку четырех значений X. Сколько различных значений может иметь $\vec{X}$? Если считать сложность построения кода пропорциональной количеству различных значений кодируемой д. с. в., то во сколько раз сложнее строить блочный код для X по сравнению с неблочным?

▶ Упражнение 24

Составить коды Хаффмена, блочный Хаффмена (для блоков длины 2 и 3) и арифметический для сообщения ABAAAB, вычислить их длины.

Приблизительный закон распределения вероятностей д. с. в., сгенерировавшей сообщение, определить анализом сообщения.

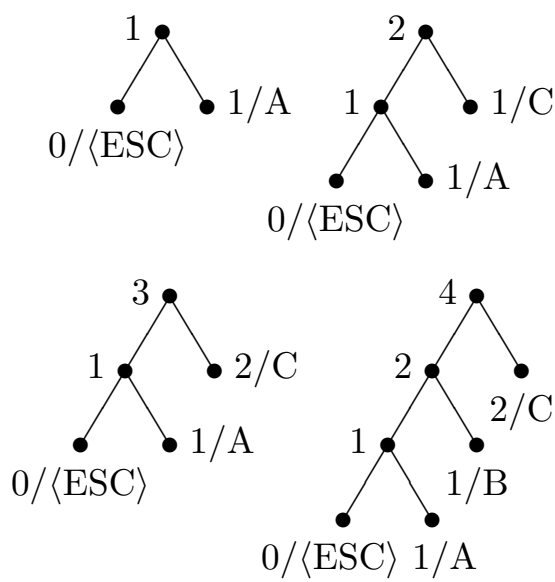
13. **Адаптивные алгоритмы сжатия. Кодирование Хаффмена**

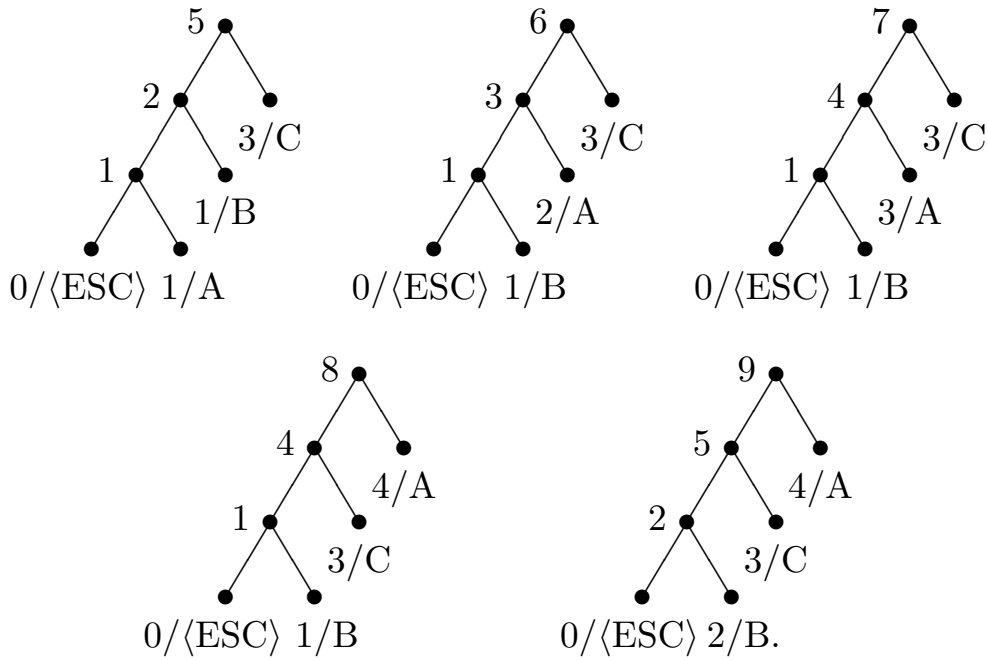
Является практичным, однопроходным, не требующим передачи таблицы кодов. Его суть в использовании *адаптивного алгоритма*, т. е. алгоритма, который при каждом сопоставлении символу кода, кроме того, изменяет внутренний ход вычислений так, что в следующий раз этому же символу может быть сопоставлен другой код, т. е. происходит адаптация алгоритма к поступающим для кодирования символам. При декодировании происходит аналогичный процесс.

В начале работы алгоритма дерево кодирования содержит только один специальный символ, всегда имеющий частоту 0. Он необходим для занесения в дерево новых символов: после него код символа передается непосредственно. Обычно такой символ называют escape-символом ($\langle$ESC$\rangle$). Расширенный ASCII кодируют каждый символ 8-битным числом, т. е. числом от 0 до 255. При построении дерева кодирования необходимо для возможности правильного декодирования как-то упорядочивать структуру дерева. Расположим листья дерева в порядке возрастания частот и затем в порядке возрастания стандартных кодов символов. Узлы собираются слева направо без пропусков. Левые ветви помечаются 0, а правые — 1.

Рассмотрим процесс построения кодов по адаптивному алгоритму Хаффмена для сообщения ACCBCAAABC, которое соответствует выборке 10-и значений д. с. в. X из 2-го примера на построение неадаптивного кода Хаффмена:

входные данные	код	длина кода	№ дерева
A	'A'	8	1
C	0'C'	9	2
C	1	1	3
B	00'B'	10	4
C	1	1	5
A	001	3	6
A	01	2	7
A	01	2	8
B	001	3	9
C	01	2	





Здесь L_1(ACCBCAAABC) = 4.1 бит/сим. Если не использовать сжатия, то L_1(ACCBCAAABC) = 8 бит/сим. Для рассматриваемой д. с. в. ранее были получены значения $ML_1(X) = 1.6$ бит/сим и $HX \approx 1.523$ бит/сим. Но с ростом длины сообщения среднее количество бит на символ сообщения при адаптивном алгоритме кодирования будет мало отличаться от значения, полученного при использовании неадаптивного метода Хаффмена или Шеннона-Фэно, т. к. алфавит символов ограничен и полный код каждого символа нужно передавать только один раз.

Теперь рассмотрим процесс декодирования сообщения 'A'0'C'100 'B'1001010100101. Здесь и далее символ в апостофах означает восемь бит, представляющих собой запись двоичного числа, номера символа, в таблице ASCII+. В начале декодирования дерево Хаффмена содержит только escape-символ с частотой 0. С раскодированием каждого нового символа дерево заново перестраивается.

входные данные	символ	№ дерева
'A'	A	1
0'C'	C	2
1	C	3
00'B'	B	4
...	...	...

Выбранный способ адаптации алгоритма очень неэффективный, т. к. после обработки каждого символа нужно перестраивать все дерево кодирования. Существуют гораздо менее трудоемкие способы, при

которых не нужно перестраивать все дерево, а нужно лишь незначительно изменять.

Бинарное дерево называется *упорядоченным*, если его узлы могут быть перечислены в порядке неубывания веса и в этом перечне узлы, имеющие общего родителя, должны находиться рядом, на одном ярусе. Причем перечисление должно идти по ярусам снизу-вверх и слева-направо в каждом ярусе.

На рис. 4 приведен пример упорядоченного дерева Хаффмена.

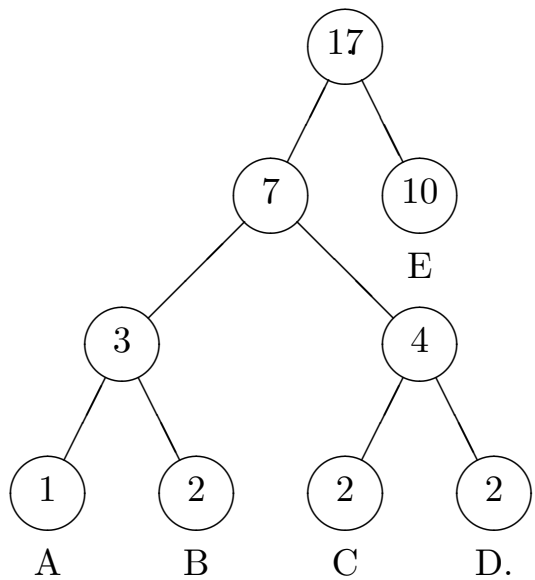


Рис. 4

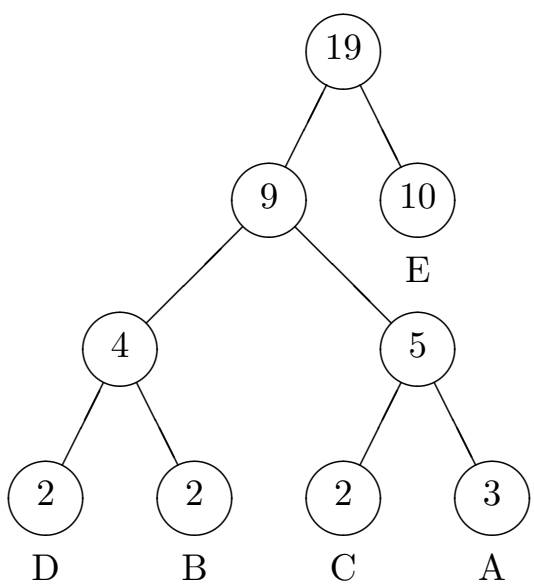


Рис. 5

Если дерево кодирования упорядоченно, то при изменении веса существующего узла дерево не нужно целиком перестраивать — в нем

достаточно лишь поменять местами два узла: узел, вес которого нарушил упорядоченность, и последний из следующих за ним узлов меньшего веса. После перемены мест узлов, необходимо пересчитать веса всех их узлов-предков.

Например, если в дереве на рис. 4 добавить еще две буквы A, то узлы A и D должны поменяться местами (см. рис. 5).

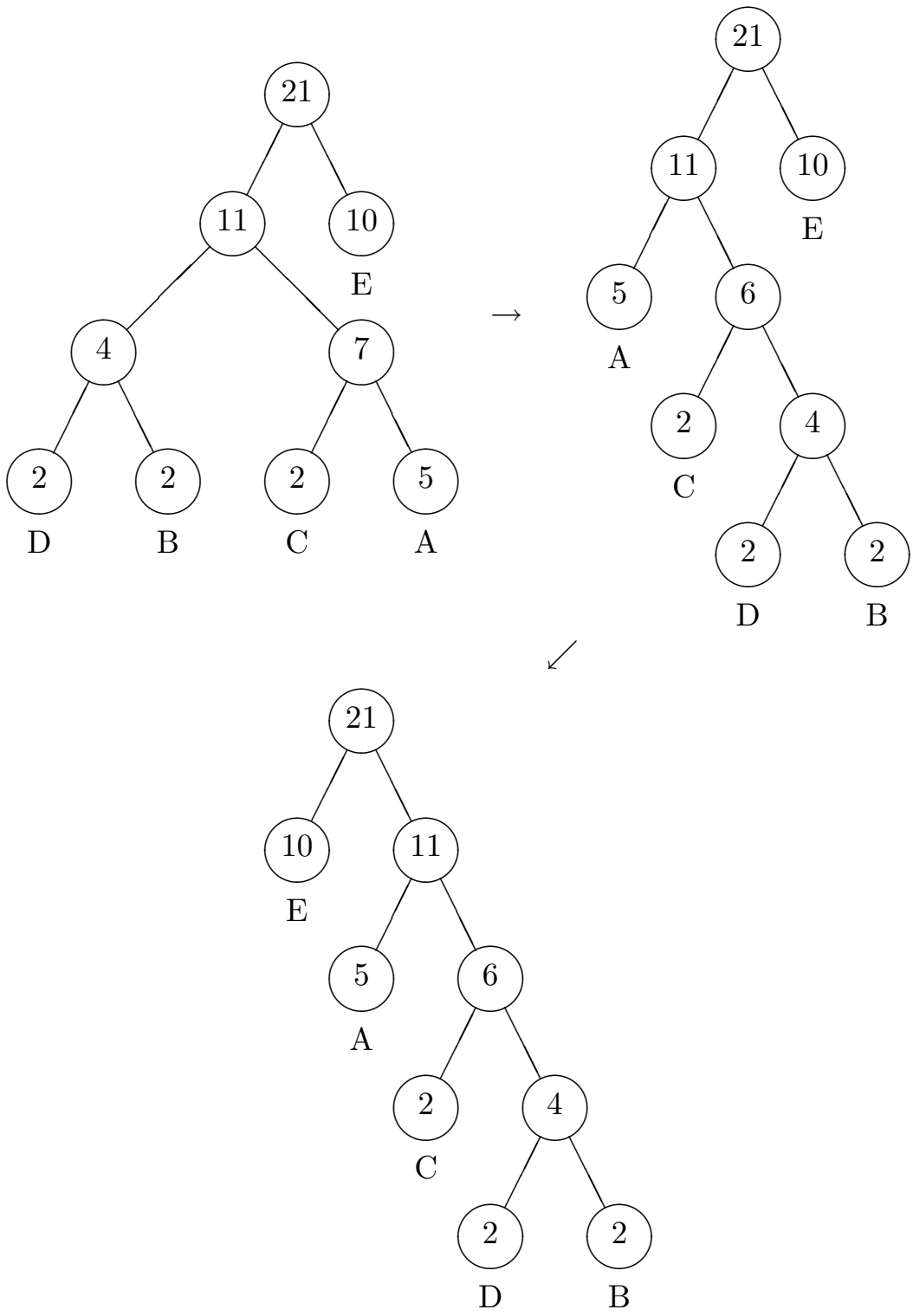


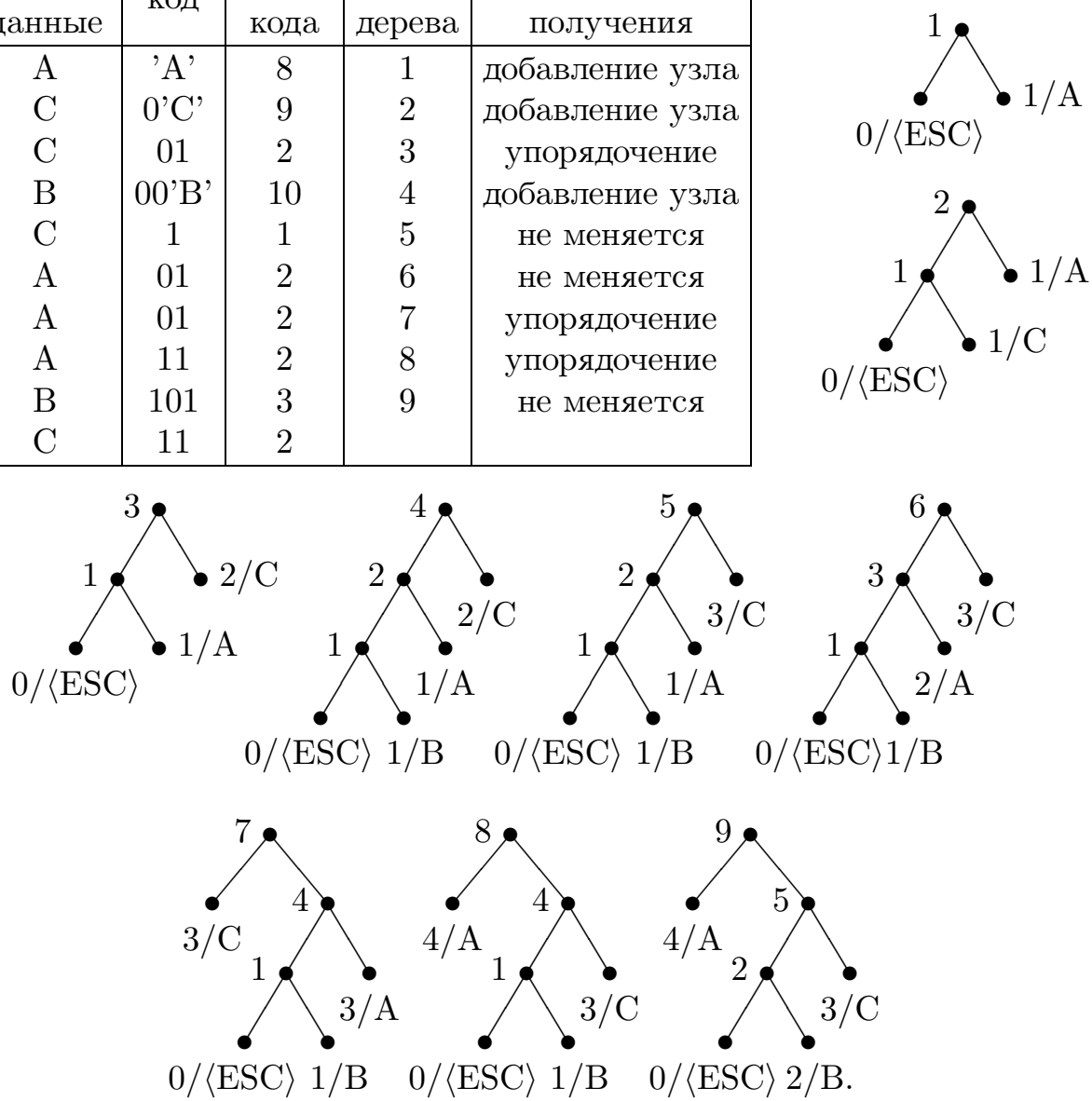
Рис. 6

Если добавить еще две буквы A, то необходимо будет поменять местами сначала узел A и узел, родительский для узлов D и B, а затем узел E и узел-брат E (рис. 6).

Дерево нужно перестраивать только при появлении в нем нового узла-листа. Вместо полной перестройки можно добавлять новый лист справа к листу ⟨ESC⟩ и упорядочивать, если необходимо, полученное таким образом дерево.

Процесс работы адаптивного алгоритма Хаффмена с упорядоченным деревом можно изобразить следующей схемой:

входные данные	код	длина кода	№ дерева	метод получения
A	'A'	8	1	добавление узла
C	0'C'	9	2	добавление узла
C	01	2	3	упорядочение
B	00'B'	10	4	добавление узла
C	1	1	5	не меняется
A	01	2	6	не меняется
A	01	2	7	упорядочение
A	11	2	8	упорядочение
B	101	3	9	не меняется
C	11	2		



Здесь получится $L_1(\text{ACCBCAAABC}) = 4.1$ бит/сим.

▶ Упражнение 25

Закодировать сообщение BBCBBC, используя адаптивный алгоритм Хаффмена с упорядоченным деревом.

▶ Упражнение 26

Закодировать сообщения “AABCDAACCCCDBB”, “КИБЕРНЕТИКИ” и “СИНЯЯ СИНЕВА СИНИ”, используя адаптивный алгоритм Хаффмена с упорядоченным деревом. Вычислить длины в битах исходного сообщения в коде ASCII+ и его полученного кода.

▶ Упражнение 27

Распаковать сообщение ’A’0’F’00’X’01111101010110111110100101, полученное по адаптивному алгоритму Хаффмена с упорядоченным деревом, рассчитать длину кода сжатого и несжатого сообщения в битах.

14. **Адаптивное арифметическое кодирование**

Для арифметического кодирования, как и для кодирования методом Хаффмена, существуют адаптивные алгоритмы. Реализация одного из них запатентована фирмой IBM.

Построение арифметического кода для последовательности символов из заданного множества можно реализовать следующим алгоритмом. Каждому символу сопоставляется его вес: вначале он для всех равен 1. Все символы располагаются в естественном порядке, например, по возрастанию. Вероятность каждого символа устанавливается равной его весу, деленному на суммарный вес всех символов. После получения очередного символа и постройки интервала для него, вес этого символа увеличивается на 1 (можно увеличивать вес любым регулярным способом).

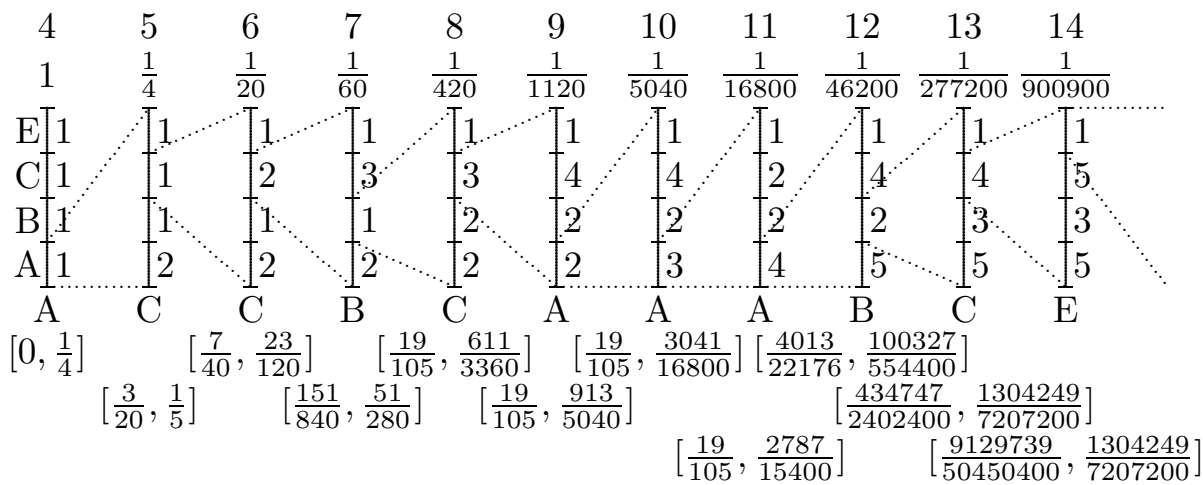


Рис. 7

Заданное множество символов — это, как правило, ASCII+. Для того, чтобы обеспечить остановку алгоритма распаковки вначале сжимаемого сообщения надо поставить его длину или ввести дополнительный символ-маркер конца сообщения. Если знать формат файла для

сжатия, то вместо начального равномерного распределения весов можно выбрать распределение с учетом этих знаний. Например, в текстовом файле недопустимы ряд управляющих символов и их вес можно занулить.

Пример. Пусть заданное множество — это символы A, B, C. Сжимаемое сообщение — ACCBCAAABC. Введем маркер конца сообщения — E. Кодирование согласно приведенному алгоритму можно провести согласно схеме, приведенной на рис. 7.

Вследствие того, что

$$\frac{759021}{2^{22} = 4194304} = 0.00101110010100111011012 \in \Big(\frac{9129739}{50450400}, \frac{1304249}{7207200}\Big),$$

$$code(\text{ACCBCAAABC}) = 001011100101001110110 1 \text{ и}$$

$$L(\text{ACCBCAAABC}) = 22.$$

Поэтому $L_1(\text{ACCBCAAABC}) = 2.2$ бит/сим. Результат, полученный адаптивным алгоритмом Хаффмена — 4.1 бит/сим, но если кодировать буквы не 8 битами, а 2, то результат будет 2.3 бит/сим. В первой строчке схемы выписаны суммарные веса символов, а во второй — длины текущих отрезков.

Способ распаковки адаптивного арифметического кода почти аналогичен приведенному для неадаптивного. Отличие только в том, что на втором шаге после получения нового кода нужно перестроить разбиение единичного отрезка согласно новому распределению весов символов. Получение маркера конца или заданного началом сообщения числа символов означает окончание работы.

Пример. Распакуем код 001011100101001110110 1, зная, что множество символов сообщения состоит из A, B, C и E, причем последний — это маркер конца сообщения.

$$0.00101110010100111011012 = \frac{759021}{4194304}.$$

Веса				Число-код и его интервал	Символ	Длина интервала
A	B	C	E			
1	1	1	1	$\frac{759021}{4194304} \in (0, \frac{1}{4})$	A	$\frac{1}{4}$
2	1	1	1	$\frac{759021}{1048576} \in (\frac{3}{5}, \frac{4}{5})$	C	$\frac{1}{5}$
2	1	2	1	$\frac{649377}{1048576} \in (\frac{1}{2}, \frac{5}{6})$	C	$\frac{1}{3}$
2	1	3	1	$\frac{375267}{1048576} \in (\frac{2}{7}, \frac{3}{7})$	B	$\frac{1}{7}$
2	2	3	1	$\frac{529717}{1048576} \in (\frac{1}{2}, \frac{7}{8})$	C	$\frac{3}{8}$
2	2	4	1	$\frac{5429}{393216} \in (0, \frac{2}{9})$	A	$\frac{2}{9}$
3	2	4	1	$\frac{16287}{262144} \in (0, 0.3)$	A	0.3
4	2	4	1	$\frac{27145}{131072} \in (0, \frac{4}{11})$	A	$\frac{4}{11}$
5	2	4	1	$\frac{298595}{524288} \in (\frac{5}{12}, \frac{7}{12})$	B	$\frac{1}{6}$
5	3	4	1	$\frac{240425}{262144} \in (\frac{8}{13}, \frac{12}{13})$	C	$\frac{4}{13}$
5	3	5	1	$\frac{1028373}{1048576} \in (\frac{13}{14}, 1)$	E	

► **Упражнение 28**
Составить адаптивный арифметический код с маркером конца для сообщения BAABC.

15. **Подстановочные или словарно-ориентированные алгоритмы сжатия информации. Методы Лемпела-Зива**

Методы Шеннона-Фэно, Хаффмена и арифметическое кодирование обобщающе называются *статистическими* методами. Словарные алгоритмы носят менее математически обоснованный, но более практичный характер.

Алгоритм LZ77 был опубликован в 1977 г. Разработан израильскими математиками Якобом Зивом (Ziv) и Авраамом Лемпелом (Lempel). Многие программы сжатия информации используют ту или иную модификацию LZ77. Одной из причин популярности алгоритмов LZ является их исключительная простота при высокой эффективности сжатия.

Основная идея LZ77 состоит в том, что второе и последующие вхождения некоторой строки символов в сообщении заменяются ссылками на ее первое вхождение.

LZ77 использует уже просмотренную часть сообщения как словарь. Чтобы добиться сжатия, он пытается заменить очередной фрагмент сообщения на указатель в содержимое словаря.

LZ77 использует “скользящее” по сообщению окно, разделенное на две неравные части. Первая, большая по размеру, включает уже просмотренную часть сообщения. Вторая, намного меньшая, является буфером, содержащим еще незакодированные символы входного потока.

Обычно размер окна составляет несколько килобайт, а размер буфера — не более ста байт. Алгоритм пытается найти в словаре (большей части окна) фрагмент, совпадающий с содержимым буфера.

Алгоритм LZ77 выдает коды, состоящие из трех элементов:

- смещение в словаре относительно его начала подстроки, совпадающей с началом содержимого буфера;

- длина этой подстроки;

- первый символ буфера, следующий за подстрокой.

Пример. Размер окна — 20 символ, словаря — 12 символов, а буфера — 8. Кодируется сообщение “ПРОГРАММНЫЕ ПРОДУКТЫ ФИРМЫ MICROSOFT”. Пусть словарь уже заполнен. Тогда он содержит строку “ПРОГРАММНЫЕ ”, а буфер — строку “ПРОДУКТЫ”. Просматривая словарь, алгоритм обнаружит, что совпадающей подстрокой будет “ПРО”, в словаре она расположена со смещением 0 и имеет длину 3 символа, а следующим символом в буфере является “Д”. Таким образом, выходным кодом будет тройка $\langle$0,3,’Д’$\rangle$. После этого алгоритм сдвигает влево все содержимое окна на длину совпадающей подстроки +1 и одновременно считывает столько же символов из входного потока в буфер. Получаем в словаре строку “РАММНЫЕ ПРОД”, в буфере — “УКТЫ ФИР”. В данной ситуации совпадающей подстроки обнаружить не удаться и алгоритм выдаст код $\langle$0,0,’У’$\rangle$, после чего сдвинет окно на один символ. Затем словарь будет содержать “АММНЫЕ ПРОДУ”, а буфер — “КТЫ ФИРМ”. И т.д.

Декодирование кодов LZ77 проще их получения, т.к. не нужно осуществлять поиск в словаре.

Недостатки LZ77:

1) с ростом размеров словаря скорость работы алгоритма-кодера пропорционально замедляется;
2) кодирование одиночных символов очень неэффективно.

Пример. Закодировать по алгоритму LZ77 строку “КРАСНАЯ КРАСКА”.

СЛОВАРЬ(8)	БУФЕР(5)	КОД
"........"	"КРАСН"	<0,0,’К’>
".......К"	"РАСНА"	<0,0,’Р’>
"......КР"	"АСНАЯ"	<0,0,’А’>
".....КРА"	"СНАЯ "	<0,0,’С’>
"....КРАС"	"НАЯ К"	<0,0,’Н’>
"...КРАСН"	"АЯ КР"	<5,1,’Я’>
".КРАСНАЯ"	" КРАС"	<0,0,’ ’>
"КРАСНАЯ "	"КРАСК"	<0,4,’К’>
"АЯ КРАСК"	"А...."	<0,0,’А’>

В последней строчке, буква “А” берется не из словаря, т.к. она

последняя.

Длина кода вычисляется следующим образом: длина подстроки не может быть больше размера буфера, а смещение не может быть больше размера словаря -1. Следовательно, длина двоичного кода смещения будет округленным в большую сторону $\log_2$(размер словаря), а длина двоичного кода для длины подстроки будет округленным в большую сторону $\log_2$(размер буфера+1). А символ кодируется 8 битами (например, ASCII+).

В последнем примере длина полученного кода равна $9*(3+3+8) = 126$ бит, против $14*8 = 112$ бит исходной длины строки.

В 1982 г. Сторером (Storer) и Шиманским (Szimanski) на базе LZ77 был разработан алгоритм LZSS, который отличается от LZ77 производимыми кодами.

Код, выдаваемый LZSS, начинается с однобитного префикса, различающего собственно код от незакодированного символа. Код состоит из пары: смещение и длина, такими же как и для LZ77. В LZSS окно сдвигается ровно на длину найденной подстроки или на 1, если не найдено вхождение подстроки из буфера в словарь. Длина подстроки в LZSS всегда больше нуля, поэтому длина двоичного кода для длины подстроки — это округленный до большего целого двоичный логарифм от длины буфера.

Пример. Закодировать по алгоритму LZSS строку “КРАСНАЯ КРАСКА”.

СЛОВАРЬ(8)	БУФЕР(5)	КОД	ДЛИНА КОДА
"........"	"КРАСН"	0’К’	9
".......К"	"РАСНА"	0’Р’	9
"......КР"	"АСНАЯ"	0’А’	9
".....КРА"	"СНАЯ "	0’С’	9
"....КРАС"	"НАЯ К"	0’Н’	9
"...КРАСН"	"АЯ КР"	1<5,1>	7
"..КРАСНА"	"Я КРА"	0’Я’	9
".КРАСНАЯ"	" КРАС"	0’ ’	9
"КРАСНАЯ "	"КРАСК"	1<0,4>	7
"НАЯ КРАС"	"КА..."	1<4,1>	7
"АЯ КРАСК"	"А...."	1<0,1>	7

Здесь длина полученного кода равна $7*9+4*7 = 91$ бит.

LZ77 и LZSS обладают следующими очевидными недостатками:

1) невозможность кодирования подстрок, отстоящих друг от друга на расстоянии, большем длины словаря;
2) длина подстроки, которую можно закодировать, ограничена размером буфера.

Если механически чрезмерно увеличивать размеры словаря и бу-

фера, то это приведет к снижению эффективности кодирования, т.к. с ростом этих величин будут расти и длины кодов для смещения и длины, что сделает коды для коротких подстрок недопустимо большими. Кроме того, резко увеличится время работы алгоритма-кодера.

В 1978 г. авторами LZ77 был разработан алгоритм LZ78, лишенный названных недостатков.

LZ78 не использует “скользящее” окно, он хранит словарь из уже просмотренных фраз. При старте алгоритма этот словарь содержит только одну пустую строку (строку длины нуль). Алгоритм считывает символы сообщения до тех пор, пока накапливаемая подстрока входит целиком в одну из фраз словаря. Как только эта строка перестанет соответствовать хотя бы одной фразе словаря, алгоритм генерирует код, состоящий из индекса строки в словаре, которая до последнего введенного символа содержала входную строку, и символа, нарушившего совпадение. Затем в словарь добавляется введенная подстрока. Если словарь уже заполнен, то из него предварительно удаляют менее всех используемую в сравнениях фразу.

Ключевым для размера получаемых кодов является размер словаря во фразах, потому что каждый код при кодировании по методу LZ78 содержит номер фразы в словаре. Из последнего следует, что эти коды имеют постоянную длину, равную округленному в большую сторону двоичному логарифму размера словаря +8 (это количество бит в байт-коде расширенного ASCII).

Пример. Закодировать по алгоритму LZ78 строку “КРАСНАЯ КРАСКА”, используя словарь длиной 16 фраз.

ВХОДНАЯ ФРАЗА (В СЛОВАРЬ)	КОД	ПОЗИЦИЯ СЛОВАРЯ
""		0
"К"	<0,’К’>	1
"Р"	<0,’Р’>	2
"А"	<0,’А’>	3
"С"	<0,’С’>	4
"Н"	<0,’Н’>	5
"АЯ"	<3,’Я’>	6
" "	<0,’ ’>	7
"КР"	<1,’Р’>	8
"АС"	<3,’С’>	9
"КА"	<1,’А’>	10

Указатель на любую фразу такого словаря — это число от 0 до 15, для его кодирования достаточно четырех бит.

В последнем примере длина полученного кода равна $10 * (4 + 8) = 120$ битам.

Алгоритмы LZ77, LZ78 и LZSS разработаны математиками и могут использоваться свободно.

В 1984 г. Уэлчем (Welch) был путем модификации LZ78 создан алгоритм LZW.

Пошаговое описание алгоритма-кодера.

Шаг 1. Инициализация словаря всеми возможными односимвольными фразами (обычно 256 символами расширенного ASCII). Инициализация входной фразы w первым символом сообщения.

Шаг 2. Считать очередной символ K из кодируемого сообщения.

Шаг 3. Если КОНЕЦ_СООБЩЕНИЯ
    Выдать код для w
    Конец
Если фраза wK уже есть в словаре
    Присвоить входной фразе значение wK
    Перейти к Шагу 2
Иначе
    Выдать код w
    Добавить wK в словарь
    Присвоить входной фразе значение K
    Перейти к Шагу 2.

Как и в случае с LZ78 для LZW ключевым для размера получаемых кодов является размер словаря во фразах: LZW-коды имеют постоянную длину, равную округленному в большую сторону двоичному логарифму размера словаря.

Пример. Закодировать по алгоритму LZW строку “КРАСНАЯ КРАСКА”. Размер словаря — 500 фраз.

ВХОДНАЯ ФРАЗА, wK (В СЛОВАРЬ)	КОД для w	ПОЗИЦИЯ СЛОВАРЯ
ASCII+		0-255
"КР"	0’К’	256
"РА"	0’Р’	257
"АС"	0’А’	258
"СН"	0’С’	259
"НА"	0’Н’	260
"АЯ"	0’А’	261
"Я "	0’Я’	262
" К"	0’ ’	263
"КРА"	<256>	264
"АСК"	<258>	265
"КА"	0’К’	266
"А"	0’А’	

В этом примере длина полученного кода равна $12 * 9 = 108$ битам.

При переполнении словаря, т. е. когда необходимо внести новую фразу в полностью заполненный словарь, из него удаляют либо наиболее редко используемую фразу, либо все фразы, отличающиеся от одиночного символа.

Алгоритм LZW является запатентованным и, таким образом, представляет собой интеллектуальную собственность. Его безлицензионное использование особенно на аппаратном уровне может повлечь за собой неприятности.

Любопытна история патентования LZW. Заявку на LZW подали почти одновременно две фирмы — сначала IBM и затем Unisys, но первой была рассмотрена заявка Unisys, которая и получила патент. Однако, еще до патентования LZW был использован в широко известной в мире Unix программе сжатия данных compress.

▶ Упражнение 29

Закодировать сообщения “AABCDAACCCCDBB”, “КИБЕРНЕТИКИ” и “СИНЯЯ СИНЕВА СИНИ”, вычислить длины в битах полученных кодов, используя алгоритмы,

LZ77 (словарь — 12 байт, буфер — 4 байта),

LZ78 (словарь — 16 фраз),

LZSS (словарь — 12 байт, буфер — 4 байта),

LZW (словарь — ASCII+ и 16 фраз).

▶ Упражнение 30

Может ли для первого символа сообщения код LZ78 быть короче кода LZW при одинаковых размерах словарей? Обосновать. Для LZW в размер словаря не включать позиции для ASCII+.

16. **LZ-алгоритмы распаковки данных. Примеры**

1. LZ77, длина словаря — 8 байт (символов). Коды сжатого сообщения — ⟨0,0,’К’⟩ ⟨0,0,’Р’⟩ ⟨0,0,’А’⟩ ⟨0,0,’С’⟩ ⟨0,0,’Н’⟩ ⟨5,1,’Я’⟩ ⟨0,0,’ ’⟩ ⟨0,4,’К’⟩ ⟨0,0,’А’⟩.

ВХОДНОЙ КОД	ПЕЧАТЬ	СЛОВАРЬ
<0,0,’К’>	"К"	".......К"
<0,0,’Р’>	"Р"	"......КР"
<0,0,’А’>	"А"	".....КРА"
<0,0,’С’>	"С"	"....КРАС"
<0,0,’Н’>	"Н"	"...КРАСН"
<5,1,’Я’>	"АЯ"	".КРАСНАЯ"
<0,0,’ ’>	" "	"КРАСНАЯ "
<0,4,’К’>	"КРАСК"	"АЯ КРАСК"
<0,0,’А’>	"А"	"Я КРАСКА"

2. LZSS, длина словаря — 8 байт (символов). Коды сжатого сообщения — 0'К' 0'Р' 0'А' 0'С' 0'Н' 1⟨5,1⟩ 0'Я' 0' ' 1⟨0,4⟩ 1⟨4,1⟩ 1⟨0,1⟩.

ВХОДНОЙ КОД	ПЕЧАТЬ	СЛОВАРЬ
0 'К'	"К"	".......К"
0 'Р'	"Р"	"......КР"
0 'А'	"А"	".....КРА"
0 'С'	"С"	"....КРАС"
0 'Н'	"Н"	"...КРАСН"
1 <5,1>	"А"	"..КРАСНА"
0 'Я'	"Я"	".КРАСНАЯ"
0 ' '	" "	"КРАСНАЯ "
1 <0,4>	"КРАС"	"НАЯ КРАС"
1 <4,1>	"К"	"АЯ КРАСК"
1 <0,1>	"А"	"Я КРАСКА"

3. LZ78, длина словаря — 16 фраз. Коды сжатого сообщения — ⟨0,'К'⟩ ⟨0,'Р'⟩ ⟨0,'А'⟩ ⟨0,'С'⟩ ⟨0,'Н'⟩ ⟨3,'Я'⟩ ⟨0,' '⟩ ⟨1,'Р'⟩ ⟨3,'С'⟩ ⟨1,'А'⟩.

ВХОДНОЙ КОД	ПЕЧАТЬ (СЛОВАРЬ)	ПОЗИЦИЯ СЛОВАРЯ
	""	0
<0,'К'>	"К"	1
<0,'Р'>	"Р"	2
<0,'А'>	"А"	3
<0,'С'>	"С"	4
<0,'Н'>	"Н"	5
<3,'Я'>	"АЯ"	6
<0,' '>	" "	7
<1,'Р'>	"КР"	8
<3,'С'>	"АС"	9
<1,'А'>	"КА"	10

4. LZW, длина словаря — 500 фраз. Коды сжатого сообщения — 0'К' 0'Р' 0'А' 0'С' 0'Н' 0'А' 0'Я' 0' ' ⟨256⟩ ⟨258⟩ 0'К' 0'А'.

При распаковке нужно придерживаться следующего правила. Словарь пополняется после считывания первого символа идущего за текущим кода, т.е. из фразы, соответствующей следующему после раскодированного коду, берется первый символ. Это правило позволяет избежать бесконечного цикла при раскодировании сообщений вида wKwK, где w — фраза, а К — символ. Конкретным примером такого сообщения является любая последовательность трех одинаковых символов, пары которых ранее не встречались.

ВХОДНОЙ КОД	ПЕЧАТЬ	СЛОВАРЬ	ПОЗИЦИЯ СЛОВАРЯ
		ASCII+	0-255
0’K’	"K"	"KP"	256
0’P’	"P"	"PA"	257
0’A’	"A"	"AC"	258
0’C’	"C"	"CH"	259
0’H’	"H"	"HA"	260
0’A’	"A"	"AЯ"	261
0’Я’	"Я"	"Я "	262
0’ ’	" "	" K"	263
<256>	"KP"	"KPA"	264
<258>	"AC"	"ACK"	265
0’K’	"K"	"KA"	266
0’A’	"A"		

► Упражнение 31
Распаковать каждое приведенное сообщение и рассчитать длину кода каждого сжатого сообщения в битах. Сообщение, сжатое LZ77 (словарь — 12 байт, буфер — 4 байта), — ⟨0,0,’A’⟩ ⟨0,0,’F’⟩ ⟨0,0,’X’⟩ ⟨9,2,’F’⟩ ⟨8,1,’F’⟩ ⟨6,2,’X’⟩ ⟨4,3,’A’⟩. Сообщение, сжатое LZSS (словарь — 12 байт, буфер — 4 байта), — 0’A’ 0’F’ 0’X’ 1⟨9,2⟩ 1⟨8,2⟩ 1⟨6,3⟩ 1⟨4,4⟩ 1⟨9,1⟩. Сообщеие, сжатое LZ78 (словарь — 16 фраз), — ⟨0,’A’⟩ ⟨0,’F’⟩ ⟨0,’X’⟩ ⟨1,’F’⟩ ⟨2,’X’⟩ ⟨5,’A’⟩ ⟨3,’A’⟩ ⟨2,’F’⟩ ⟨0,’A’⟩. Сообщение, сжатое LZW (словарь — ASCII+ и 16 фраз), — 0’A’ 0’F’ 0’X’ ⟨256⟩ ⟨257⟩ ⟨257⟩ 0’A’ ⟨258⟩ 0’F’ 0’F’ 0’A’.

17. **Особенности программ-архиваторов**

Если коды алгоритмов типа LZ передать для кодирования (адаптивному) алгоритму Хаффмена или арифметическому, то полученный двухшаговый (конвейерный, а не двухпроходный) алгоритм даст результаты сжатия подобные широко известным программам GZIP, ARJ, PKZIP и подобным.

Наибольшую степень сжатия дают двухпроходные алгоритмы, которые исходные данные последовательно сжимают два раза, но они работают до двух раз медленнее однопроходных при незначительном увеличении степени сжатия.

Большинство программ-архиваторов сжимает каждый файл по отдельности, но некоторые сжимают файлы в общем потоке, что дает увеличение степени сжатия, но одновременно усложняет способы работы с полученным архивом, например, замена в таком архиве файла на его более новую версию может потребовать перекодирования всего архива. Примером программы, имеющей возможность сжимать файлы в общем

потоке, является RAR. Архиваторы ОС Unix (gzip, bzip2, ...) сжимают файлы в общем потоке практически всегда.

В 1992 году фирма WEB Technologies объявила о выходе новой программы сжатия DataFiles/16, которая якобы может при неоднократном использовании сжать любое количество данных до 1024 байт. Информация об этом прошла из солидного издания, журнала Byte.

Конечно же никакой алгоритм сжатия не может уплотнить произвольные данные. Для доказательства этого проделаем следующий мысленный эксперимент. Предположим, что на жестком диске компьютера хранятся все возможные разные файлы длиной ровно 100 байт (таких файлов будет всего 2^{800}). И пусть существует идеальная программа сжатия данных, которая сожмет каждый из них хотя бы на один байт. Но тогда, так как всего разных файлов длиной меньшей 100 байт существует не более чем $1 + 2^8 + 2^{16} + \cdots + 2^{792} = (2^{800} - 1)/255 < 2^{800}$, то неизбежно получится, что два разных файла упакуются в идентичные файлы. Следовательно, не может существовать программы сжатия данных, которая может сжать любые исходные данные.

Формат файла, содержащего данные, которые перед использованием требуется распаковать соответствующей программой-архиватором, как правило, может быть идентифицирован расширением имени файла.

В следующей таблице приводятся некоторые типичные расширения, соответствующие им программы-архиваторы и методы сжатия данных.

Расширения	Программы	Тип кодирования
Z	compress	LZW
arc	arc, pkarc	LZW, Хаффмена
zip	zip, unzip, pkzip, pkunzip	LZW, LZ77, Хаффмена, Шеннона-Фэно
gz	gzip	LZ77, Хаффмена
bz2	bzip2	Берроуза-Уиллера, Хаффмена
arj	arj	LZ77, Хаффмена
ice, lzh	lha, lharc	LZSS, Хаффмена
pak	pak	LZW

Практически все форматы файлов для хранения графической информации используют сжатие данных. Формат графического файла также, как правила, идентифицируется расширением имени файла.

В следующей таблице приводятся некоторые типичные расширения графических файлов и соответствующие им методы сжатия данных.

Расширения	Тип кодирования
gif	LZW
jpeg, jpg	сжатие с потерями, Хаффмена или арифметическое
bmp, pcx	RLE
tiff, tif	CCITT/3 для факсов, LZW или другие

Сжатие RLE (Run Length Encoding — кодирование переменной длины) — это простейший метод сжатия, в общем случае очень неэффективный, но дающий неплохие результаты на типичной графической информации. Оно основано в основном на выделении специального кода-маркера, указывающего сколько раз повторить следующий байт.

Сжатие и распаковка в реальном времени используется в программах-драйверах для "уплотнения" носителей информации, позволяющих увеличить емкость носителя приблизительно в 2 раза. Наиболее известной программой такого рода является DriverSpace для MS-DOS и Microsoft Windows.

18. **Сжатие информации с потерями**

Все ранее рассмотренные алгоритмы сжатия информации обеспечивали возможность полного восстановления исходных данных. Но иногда для повышения степени сжатия можно отбрасывать часть исходной информации, т. е. производить сжатие с потерями. Естественно, что такое сжатие нельзя проводить, например, на финансовой базе данных банка. Но в тех случаях, когда сжимается информация, используемая лишь для качественной оценки (это, как правило, аналоговая информация), сжатие с потерями является очень подходящим.

Сжатие с потерями используется в основном для трех видов данных: полноцветная графика ($2^{24} \approx 16$ млн. цветов), звук и видеоинформация.

Сжатие с потерями обычно проходит в два этапа. На первом из них исходная информация приводится (с потерями) к виду, в котором ее можно эффективно сжимать алгоритмами 2-го этапа сжатия без потерь.

Основная идея сжатия графической информации с потерями заключается в следующем. Каждая точка в картинке характеризуется тремя равноважными атрибутами: яркостью, цветом и насыщенностью. Но глаз человека воспринимает эти атрибуты не как равные. Глаз воспринимает полностью только информацию о яркости и в гораздо меньшей степени о цвете и насыщенности, что позволяет отбрасывать часть информации о двух последних атрибутах без потери качества изображения. Это свойство зрения используется, в частности, в цветном телевизоре, в котором на базовое черно-белое изображение наносят цветовую раскраску.

Для сжатия графической информации с потерями в конце 1980-х установлен один стандарт — формат JPEG (Joint Photographic Experts Group — название объединения его разработчиков). В этом формате можно регулировать степень сжатия, задавая степень потери качества.

Сжатие видеоинформации основано на том, что при переходе от одного кадра фильма к другому на экране обычно почти ничего не меняется. Таким образом, сжатая видеоинформация представляет собой запись некоторых базовых кадров и последовательности изменений в них. При этом часть информации может отбрасываться. Сжатую подобным образом информацию можно далее сжимать и другими методами. Хотя существует не один стандарт для сжатия видеоданных, наиболее распространенными являются стандарты MPEG (Motion Picture Experts Group), первый из которых был опубликован в 1988 году. MPEG — практически единственный стандарт для записи видео и звуковой информации на CD-ROM, DVD-ROM и в цифровом спутниковом телевидении. Видеоинформацию можно сжать необыкновенно плотно, до 100 и более раз, что позволяет, например, на одну видеокассету, записать более ста различных художественных фильмов. Но из-за очень сложных проблем, связанных с правами на интеллектуальную собственность, реально возможности сжатия информации таким образом используются сравнительно редко.

Для сжатии звуковой информации с потерями существует несколько стандартов. Наиболее широко используемый из них — это MPEG без видеоданных. Стандарт LPC (Linear Predictive Coding) используется для сжатия речи. Алгоритм LPC пытается промоделировать речевой тракт человека и выдает на выходе буквально текущее состояние участвующих в формировании звуков органов.

19. **Информационный канал**

Канал информационный — это совокупность устройств, объединенных линиями связи, предназначенных для передачи информации от источника информации (*начального устройства канала*) до ее приемника (*конечного устройства канала*).

Линии связи обеспечивают прохождение информационных сигналов между устройствами канала. Информация обычно передается при помощи электрического тока (по проводам), света (по оптоволокну), электромагнитных волн радиодиапазона (в пространстве) и, редко, звука (в плотной среде: атмосфере, воде и т. п.) и прочих.

Устройства канала — это, как правило, *репитеры*, просто передающие усиленным принятый сигнал (пример, радиорелейные линии). К устройствам канала иногда относят и кодеры/декодеры, но в только тех случаях, когда кодирование/декодирование происходит с высокой скоростью, не требующей ее специального учета, как замедляющего

фактора; обычно же кодеры/декодеры относят к источникам или приемникам информации.

Технические характеристики канала определяются принципом действия входящих в него устройств, видом сигнала, свойствами и составом физической среды, в которой распространяются сигналы, свойствами применяемого кода.

Эффективность канала характеризуется скоростью и достоверностью передачи информации, надежностью работы устройств и задержкой сигнала во времени.

Задержка сигнала во времени — это интервал времени от отправки сигнала передатчиком до его приема приемником.

Математически канал задается множеством допустимых сообщений на входе, множеством допустимых сообщений на выходе и набором условных вероятностей $P(y/x)$ получения сигнала y на выходе при входном сигнале x. Условные вероятности описывают статистические свойства "шумов" (или помех), искажающих сигнал в процессе передачи. В случае, когда $P(y/x) = 1$ при $y = x$ и $P(y/x) = 0$ при $y \neq x$, канал называется *каналом без "шумов"*. В соответствии со структурой входных и выходных сигналов выделяют дискретные и непрерывные каналы. В дискретных каналах сигналы на входе и выходе представляют собой последовательность символов одного или двух (по одному для входа и выхода) алфавитов. В непрерывных каналах входной и выходной сигналы представляют собой функции от непрерывного параметра-времени. Бывают также смешанные или гибридные каналы, но тогда обычно рассматривают их дискретные и непрерывные компоненты раздельно. Далее рассматриваются только дискретные каналы.

Способность канала передавать информацию характеризуется числом — *пропускной способностью* или *емкостью канала* (обозначение — C).

Для случая канала без шума формула расчета емкости канала имеет вид $C = \lim\limits_{T\to\infty} \frac{\log_2 N(T)}{T}$, где $N(T)$ — число всех возможных сигналов за время T.

Пример. Пусть алфавит канала без "шумов" состоит из двух символов — 0 и 1, длительность τ секунд каждый. За время T успеет пройти $n = T/\tau$ сигналов, всего возможны 2^n различных сообщений длиной n. В этом случае $C = \lim\limits_{T\to\infty} \frac{\log_2 2^{T/\tau}}{T} = 1/\tau$ бод.

На рис. 8 приведена схема, на которой изображен процесс прохождения информации по каналу с описанными в примере характеристиками.

Здесь для кодирования используется уровень сигнала: низкий для 0 и высокий для 1. Недостатки этого способа проявляются в случаях, когда нужно передавать много сплошных нулей или единиц. Малей-

шее рассогласование синхронизации между приемником и передатчиком приводит тогда к неисправимым ошибкам. Кроме того, многие носители информации, в частности, магнитные, не могут поддерживать длительный постоянный уровень сигнала.

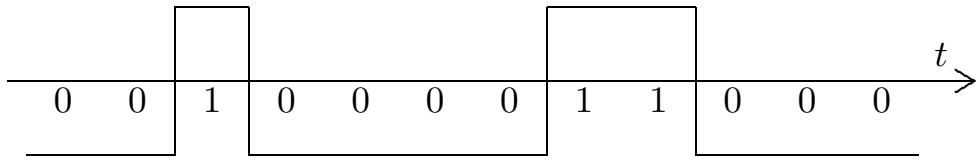


Рис. 8

Для передачи информации используется обычно другой способ, когда для представления 0 и 1 используются две разные частоты, отличающиеся друг от друга ровно в два раза (см. рис. 9) — это так называемая *частотная модуляция* (ЧМ или FM).

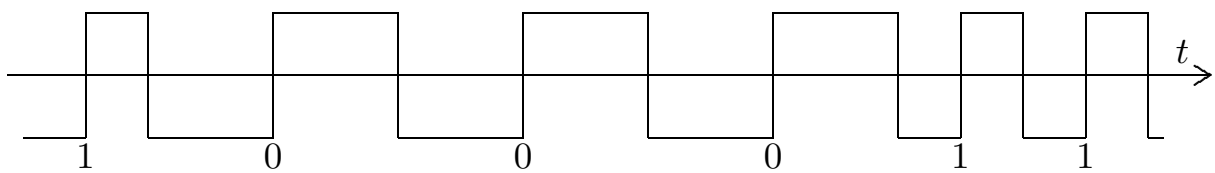


Рис. 9

Таким образом, при таком кодировании, если сигнал 1 имеет длительность τ, то 0 — 2τ.

Рассчитаем емкость этого канала. Нужно рассчитать $N(T)$. Пусть $n = T/\tau$, тогда получается, что нужно рассчитать сколькими способами можно разбить отрезок длины n отрезками длины 2 и 1. Получаем, что $N(T) = S_n = C_n^n + C_{n-1}^{n-2} + C_{n-2}^{n-4} + \cdots$, где первое слагаемое — это количество способов, которыми можно разбить отрезок длины n n отрезками длины 1, второе слагаемое — это количество способов, которыми можно разбить отрезок длины n $(n-2)$ отрезками длины 1 и одним отрезком длины 2, третье слагаемое — это количество способов, которыми можно разбить отрезок длины n $(n-4)$ отрезками длины 1 и двумя отрезками длины 2 и т.д. Таким образом, $S_1 = 1$. Вследствие того, что $C_m^k + C_m^{k+1} = C_{m+1}^{k+1}$ для любых $k < m$, получается, что

$$\begin{array}{rcccccccccc} S_{n-1} &=& & & C_{n-1}^{n-1} &+& C_{n-2}^{n-3} &+& C_{n-3}^{n-5} &+& \cdots \\ S_n &=& C_n^n &+& C_{n-1}^{n-2} &+& C_{n-2}^{n-4} &+& C_{n-3}^{n-6} &+& \cdots, \\ S_{n+1} &=& C_{n+1}^{n+1} &+& C_n^{n-1} &+& C_{n-1}^{n-3} &+& C_{n-2}^{n-5} &+& \cdots \end{array}$$

т. е. $S_{n+1} = S_n + S_{n-1}$ при $n > 1$. Если положить, что $S_0 = 1$, то $S_0, S_1, \ldots$ — это последовательность $1, 1, 2, 3, 5, 8, 13, 21, 34, \ldots$, т.е. числа Фибоначчи. С XIX века для вычисления n-го члена последовательности Фибоначчи известна формула

$$S_n = \frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^{n+1} - \left(\frac{1-\sqrt{5}}{2}\right)^{n+1}\right).$$

Таким образом, $C = \lim\limits_{T\to\infty} \frac{\log_2 N(T)}{T} = \lim\limits_{n\to\infty} \frac{\log_2 S_n}{n\tau} = \frac{\log_2 \frac{1+\sqrt{5}}{2}}{\tau} \approx 0.69/\tau$ бод.

При использовании частотной модуляции на практике нули, как правило, кодируются в два раза плотнее. Это достигается тем, что учитываются не уровни сигнала, а смена уровня (полярности). Если частота ν соответствует 1, то с частотой 2ν производится проверка уровня сигнала. Если он меняется, то это сигнал 1, если нет, то — 0. На практике частота ν — это частота синхронизации, т. е. частота импульса, который независимо от данных меняет полярность сигнала. 0 не генерирует импульса смены полярности, а 1 генерирует (см. рис. 10).

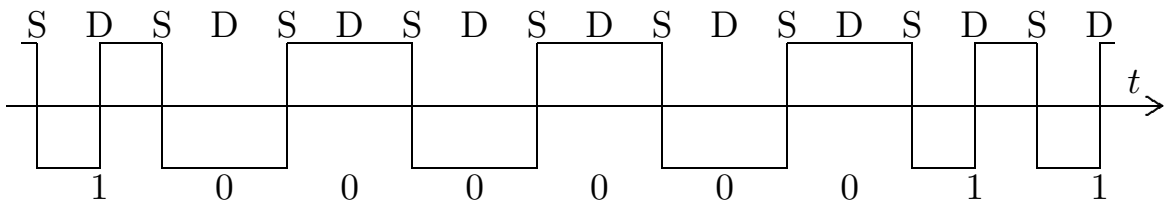


Рис. 10

Для записи информации на первые магнитные диски и ленты использовался метод FM. На гибкие диски 5.25” и 3.5” информация записывается методом MFM (Modified FM) — модификацией метода FM, позволяющей в 2 раза повысить плотность записи. Это достигается тем, что частота синхронизации увеличивается вдвое. MFM можно использовать с теми же физическими каналами, что и FM, потому что импульсы синхронизации не передаются перед 1 и первым 0 в серии нулей (см. рис. 11).

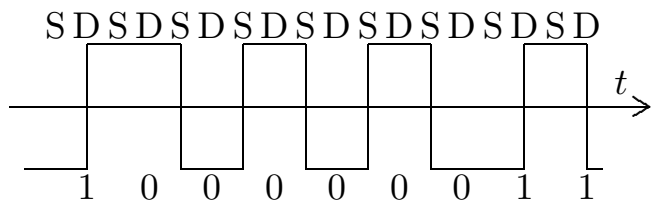


Рис. 11

Метод записи с групповым кодированием, RLL — Run Limited Length, не использует импульсы синхронизации, применяется, в частности, в жестких дисках “винчестер” и существует в нескольких разновидностях. Одна из них основана на замене тетрад байта на 5-битные группы. Эти группы подбираются таким образом, чтобы при передаче данных нули не встречались подряд более двух раз, что делает код самосинхронизирующимся. Например, тетрада 1000 заменяется группой бит 11010, а тетрада 0001 — 11011 (см. рис. 12). Такое кодирование позволяет без увеличения частоты сигнала в среднем на несколько процентов повысить скорость передачи данных по сравнению с MFM. Суще-

ствуют разновидности RLL, в которых заменяются последовательности бит различной длины. Кодирование MFM или FM можно представить как частный случай RLL.

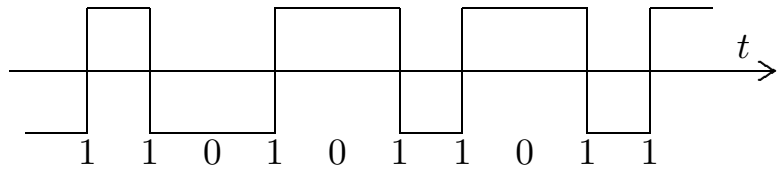


Рис. 12

При необходимости передачи записанных с помощью некоторого кода сообщений по данному каналу приходиться преобразовывать эти сообщения в допустимые сигналы канала, т. е. производить надлежащее кодирование, а при приеме данных — декодирование. Кодирование целесообразно производить так, чтобы среднее время, затрачиваемое на передачу, было как можно меньше. Получается, что исходному входному алфавиту нужно однозначно сопоставить новый алфавит, обеспечивающий большую скорость передачи.

Следующий, *основной факт теории передачи информации* или *основная теорема о кодировании при наличии помех* позволяет при знании емкости канала и энтропии передатчика вычислить максимальную скорость передачи данных в канале.

Теорема Шеннона. Пусть источник характеризуется д. с. в. X. Рассматривается канал с шумом, т. е. для каждого передаваемого сообщения задана вероятность ε его искажения в процессе передачи (вероятность ошибки). Тогда существует такая скорость передачи u, зависящая только от X, что $\forall \varepsilon > 0\ \exists u' < u$ сколь угодно близкая к u такая, что существует способ передавать значения X со скоростью u' и с вероятностью ошибки меньшей ε, причем $u = C/HX$. Упомянутый способ образует помехоустойчивый код.

Кроме того, Фэно доказана [20] следующая *обратная теорема о кодировании при наличии помех*. Для $u' > u$ можно найти такое положительное число ε, что в случае передачи информации по линии связи со скоростью u' вероятность ошибки ε передачи каждого символа сообщения при любом методе кодирования и декодирования будет не меньше ε (ε очевидно растет вслед за ростом u').

► Упражнение 32

По каналу связи без шума могут передаваться четыре сигнала длительностью 1 мс каждый. Вычислить емкость такого канала.

► Упражнение 33

Три передатчика задаются случайными величинами со следующими законами распределениями вероятностей:

1) $P(X_1 = -1) = 1/4$, $P(X_1 = 0) = 1/2$, $P(X_1 = 1) = 1/4$;

2) $P(X_2 = -1) = 1/3$, $P(X_2 = 0) = 1/3$, $P(X_2 = 1) = 1/3$;
3) $P(X_3 = n) = 2^{-n}$, $n = 1, 2, \ldots$

Емкость канала связи с шумом равна 4000 бод. Вычислить максимальную скорость передачи данных по этому каналу каждым из передатчиков, обеспечивающую сколь угодно высокую надежность передачи.

20. **Помехозащитное кодирование**

Простейший код для борьбы с шумом — это контроль четности, он, в частности, широко используется в модемах. Кодирование заключается в добавлении к каждому байту девятого бита таким образом, чтобы дополнить количество единиц в байте до заранее выбранного для кода четного (even) или нечетного (odd) значения. Используя этот код, можно лишь обнаруживать большинство ошибок.

Простейший код, исправляющий ошибки, — это тройное повторение каждого бита. Если с ошибкой произойдет передача одного бита из трех, то ошибка будет исправлена, но если случится двойная или тройная ошибка, то будут получены неправильные данные. Часто коды для исправления ошибок используют совместно с кодами для обнаружения ошибок. При тройном повторении для повышения надежности три бита располагают не подряд, а на фиксированном расстоянии друг от друга. Использование тройного повторения значительно снижает скорость передачи данных.

Двоичный симметричный канал изображен на рис. 13, где p — это вероятность безошибочной передачи бита, а q — вероятность передачи бита с ошибкой. Предполагается, что в таком канале ошибки происходят независимо. Далее рассматриваются только такие каналы.

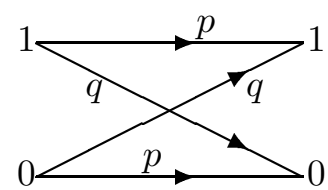


Рис. 13

Двоичный симметричный канал реализует схему Бернулли, поэтому вероятность передачи n бит по двоичному симметричному каналу с k ошибками равна $P_n(k) = C_n^k p^{n-k} q^k$.

Пример. Вероятность передачи одного бита информации с ошибкой равна $q = 0.01$ и нас интересует вероятность безошибочной передачи 1000 бит (125 байт). Искомую вероятность можно подсчитать по формуле $P_{1000}(0) = C_{1000}^0 p^{1000} q^0 = 0.99^{1000} \approx 4.32 * 10^{-5}$, т.е. она ничтожно мала.

Добиться минимальности вероятности ошибки при передаче данных можно используя специальные коды. Обычно используют *систематические* помехозащитные коды. Идея систематических кодов состоит в добавлении к символам исходных кодов, предназначенных для передачи в канале, нескольких контрольных символов по определенной схеме кодирования. Принятая такая удлиненная последовательность кодов декодируется по схеме декодирования в первоначально переданную.

Приемник способен распознавать и/или исправлять ошибки, вызванные шумом, анализируя дополнительную информацию, содержащуюся в удлиненных кодах.

Двоичным (m, n)*-кодом* называется пара, состоящая из схемы кодирования $E\colon \mathbb{Z}_2^m \to \mathbb{Z}_2^n$ и схемы декодирования $D\colon \mathbb{Z}_2^n \to \mathbb{Z}_2^m$, где $\mathbb{Z}_2^n$ — множество всех двоичных последовательностей длины n, $m < n$ (случай $m = n$ рассматривается в криптографии).

Функции D и E выбираются так, чтобы функция $H = D \circ T \circ E$, где T — *функция ошибок*, с вероятностью, близкой к единице, была тождественной. Функции D и E считаются безошибочными, т. е. функция $D \circ E$ — тождественная (см. рис. 14).

	E		T		D	
Исходное сообщение	$\longrightarrow$	Кодированное сообщение	$\longrightarrow$	Двоичный симметричный канал	$\longrightarrow$	Принятое сообщение $\longrightarrow$ Декодированное сообщение

Рис. 14

▶ Упражнение 34
Пусть двоичный симметричный канал используется для передачи строк из двух бит. Построить таблицу вероятностей приема.

▶ Упражнение 35
По двоичному симметричному каналу передаются строки длины 14. Какова вероятность того, что ровно пять символов будут приняты неправильно? Какова вероятность того, что менее пяти символов будут приняты неправильно? Сколько имеется строк, отличающихся от данной не больше, чем в четырех позициях?

21. Математическая модель системы связи

Коды делятся на два больших класса. *Коды с исправлением ошибок* имеют целью восстановить с вероятностью, близкой к единице, посланное сообщение. *Коды с обнаружением ошибок* имеют целью выявить с вероятностью, близкой к единице, наличие ошибок.

Простой код с обнаружением ошибок основан на схеме *проверки четности*, применимой к сообщениям $a_1 \dots a_m$ любой фиксированной длины m. Схема кодирования определяется следующими формулами:

$$E(a_1 \dots a_m) = a_1 \dots a_m a_{m+1},$$

$$a_{m+1} = \begin{cases} 0, & \text{если } \sum_{i=1}^m a_i \text{ — четная;} \\ 1, & \text{если } \sum_{i=1}^m a_i \text{ — нечетная.} \end{cases}$$

Таким образом, $\sum_{i=1}^{m+1} a_i$ должна быть четной.

Соответствующая схема декодирования тривиальна:

$$D(a_1 \dots a_m a_{m+1}) = \begin{cases} a_1 \dots a_m, & \text{если } \sum_{i=1}^{m+1} a_i \text{ — четна;} \\ \langle \textit{ошибка} \rangle, & \text{если } \sum_{i=1}^{m+1} a_i \text{ — нечетна.} \end{cases}$$

Разумеется, что четность $\sum_{i=1}^{m+1} a_i$ не гарантирует безошибочной передачи.

Пример. Проверка четности при $m = 2$ реализуется следующим кодом (функцией E): $00 \to 000$, $01 \to 011$, $10 \to 101$, $11 \to 110$. В двоичном симметричном канале доля неверно принятых сообщений для этого кода (хотя бы с одной ошибкой) равна $q^3 + 3pq^2 + 3p^2q$ (три, две или одна ошибка соответственно). Из них незамеченными окажутся только ошибки точно в двух битах, не изменяющие четности. Вероятность таких ошибок $3pq^2$. Вероятность ошибочной передачи сообщения из двух битов равна $2pq + q^2$. При малых q верно, что $3pq^2 \ll 2pq + q^2$.

Рассмотрим $(m, 3m)$-код с *тройным повторением*. Коды с повторениями очень неэффективны, но полезны в качестве теоретического примера кодов, исправляющих ошибки. Любое сообщение разбивается на блоки длиной m каждое и каждый блок передается трижды — это определяет функцию E. Функция D определяется следующим образом. Принятая строка разбивается на блоки длиной $3m$. Бит с номером i $(1 \leqslant i \leqslant m)$ в декодированном блоке получается из анализа битов с номерами i, $i+m$, $i+2m$ в полученном блоке: берется тот бит из трех, который встречается не менее двух раз. Вероятность того, что бит в данной позиции будет принят трижды правильно равна p^3. Вероятность одной ошибки в тройке равна $3p^2q$. Поэтому вероятность правильного приема одного бита равна $p^3 + 3p^2q$. Аналогичным образом получается, что вероятность приема ошибочного бита равна $q^3 + 3pq^2$.

Пример. Предположим $q = 0.1$. Тогда вероятность ошибки при передачи одного бита — 0.028, т. е. этот код снижает вероятность ошибки с 10% до 2.8%. Подобным образом организованная передача с пятикратным повторением даст вероятность ошибки на бит

$$q^5 + 5pq^4 + 10p^2q^3 = 0.00856 = 0.856\%,$$

т. е. менее 1%. В результате вероятность правильной передачи строки длиной 10 возрастет с $0.9^{10} \approx 35\%$ до $0.972^{10} \approx 75\%$ при тройных повторениях и до $0.99144^{10} \approx 92\%$ при пятикратных повторениях.

Тройное повторение обеспечивает исправление одной ошибки в каждой позиции за счет трехкратного увеличения времени передачи.

Рассмотрим $(2048, 2313)$-код, используемый при записи данных на магнитофонную ленту компьютерами Apple II. К каждому байту исходных данных прибавляется бит четности и, кроме того, после каждых таких расширенных битом четности 256 байт добавляется специальный байт, также расширенный битом четности. Этот специальный

байт, который называют *контрольной суммой* (check sum), есть результат применения поразрядной логической операции “исключающее ИЛИ” (XOR) к 256 предшествующим расширенным байтам. Этот код способен как обнаруживать ошибки нечетной кратности в каждом из отдельных байтов, так и исправлять до 8 ошибок в блоке длиной 256 байт. Исправление ошибок основано на том, что если в одном из бит одного из байт 256 байтового блока произойдет сбой, обнаруживаемый проверкой четности, то этот же сбой проявится и в том, что результат операции “исключающее ИЛИ” над всеми соответствующими битами блока не будет соответствовать соответствующему биту контрольной суммы. Сбойный бит однозначно определяется пересечением сбойных колонки байта и строки бита контрольной суммы. На рис. 15 изображена схема участка ленты, содержащего ровно 9 ошибок в позициях, обозначенных p_1, p_2, …, p_9. Расширенный байт контрольной суммы обозначен CS, а бит паритета (в данном случае четности) — PB (parity bit). Ошибка в позиции p_1 может быть исправлена. Ошибки в позициях p_4, p_5, p_6, p_7 можно обнаружить, но не исправить. Ошибки в позициях p_2, p_3, p_8, p_9 невозможно даже обнаружить.

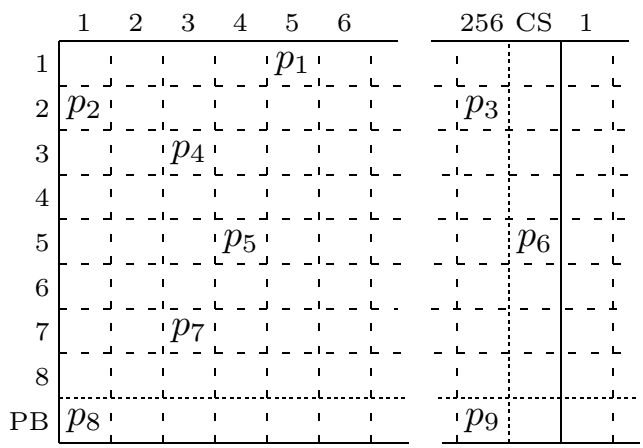


Рис. 15

Приведенные ранее примеры простейших кодов принадлежат к классу *блочных*. По определению, блочный код заменяет каждый блок из m символов более длинным блоком из n символов. Следовательно, (m, n)-коды являются блочными. Существуют также *древовидные* или *последовательные коды*, в которых значение очередного символа зависит от всего предшествующего фрагмента сообщения. Работа с древовидным шумозащитным кодом имеет сходство с работой с арифметическим кодом для сжатия информации.

Расстоянием (Хэмминга) между двоичными словами длины n называется количество позиций, в которых эти слова различаются. Это одно из ключевых понятий теории кодирования. Если обозначить двоичные слова как $a = a_1 \dots a_n$ и $b = b_1 \dots b_n$, то расстояние между ними обозначается $d(a, b)$.

Весом двоичного слова $a = a_1 \dots a_n$ называется количество единиц в нем. Обозначение $w(a)$. Можно сказать, что $w(a) = \sum_{i=1}^{n} a_i$.

Пример. Пусть $a = 1001$ и $b = 0011$, тогда $w(a) = w(b) = 2$, $d(a, b) = 2$.

Далее операция $+$ при применении к двоичным словам будет означать поразрядное сложение без переноса, т. е. сложение по модулю 2 или "исключающее ИЛИ" (XOR).

Расстояние между двоичными словами a и b равно весу их поразрядной суммы, т. е. $d(a, b) = w(a + b)$.

Если два слова различаются в каком-либо разряде, то это добавит единицу к весу их поразрядной суммы.

Следовательно, если a и b — слова длины n, то вероятность того, что слово a будет принято как b, равна $p^{n-d(a,b)}q^{d(a,b)}$.

Наример, вероятность того, что слово 1011 будет принято как 0011, равна p^3q.

Для возможности обнаружения ошибки в одной позиции минимальное расстояние между словами кода должно быть бóльшим 1.

Иначе ошибка в одной позиции сможет превратить одно кодовое слово в другое, что не даст ее обнаружить.

Для того, чтобы код давал возможность обнаруживать все ошибки кратности, не большей k, необходимо и достаточно, чтобы наименьшее расстояние между его словами было $k + 1$.

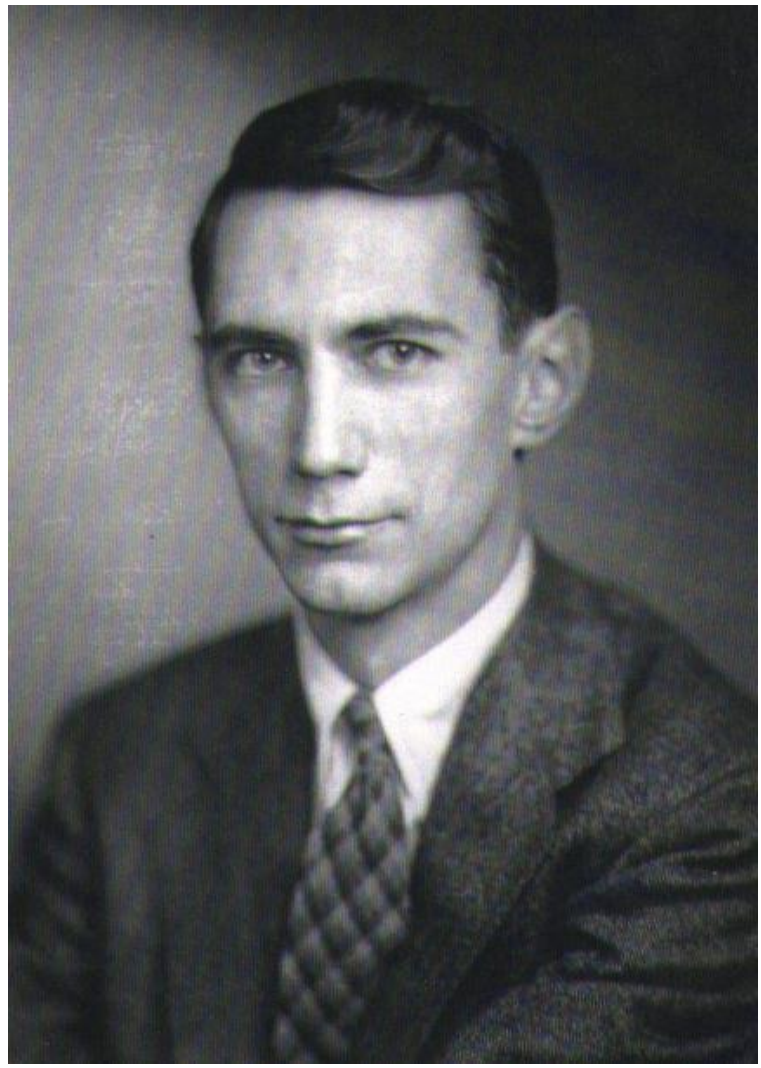
Достаточность доказывается конструктивно: если условие утверждения выполнено на E, то в качестве декодирующей функции D следует взять функцию, сообщающую об ошибке, если декодируемое слово отличается от любого из слов из образа E. Необходимость доказывается от противного: если минимальное расстояние $k' < k + 1$, то ошибка в k' позициях сможет превратить одно кодовое слово в другое.

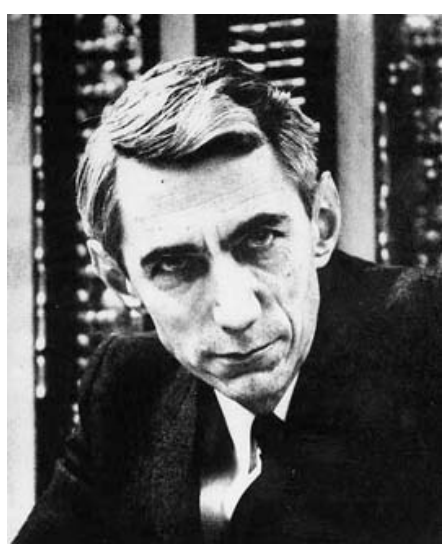
Для такого кода вероятность того, что ошибки в сообщении останутся необнаруженными, равна

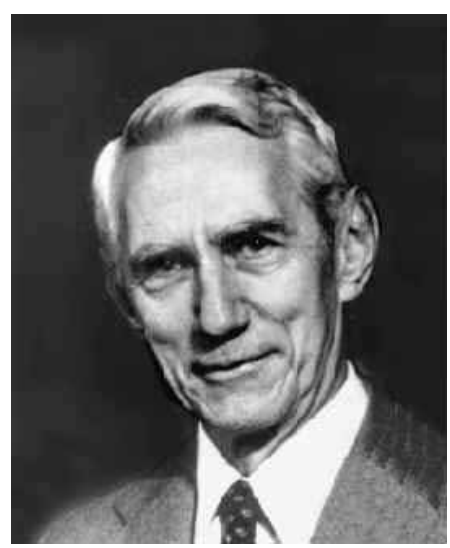
$$\sum_{i=k+1}^{n} C_n^i p^{n-i} q^i = C_n^{k+1} p^{n-k-1} q^{k+1} + \dots + C_n^{n-1} p q^{n-1} + q^n \approx$$

$\approx$ [при малых q и не слишком маленьких k] $\approx C_n^{k+1} p^{n-k-1} q^{k+1}$.

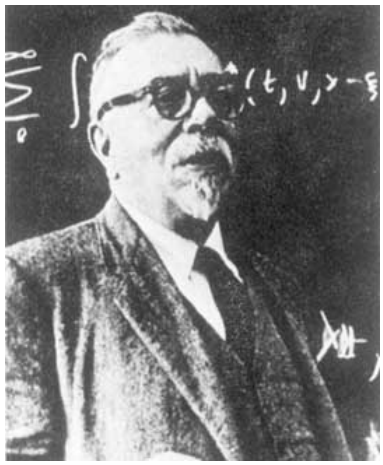
Для того чтобы код давал возможность исправлять все ошибки кратности, не большей k, необходимо и достаточно, чтобы наименьшее расстояние между его словами было $2k + 1$.



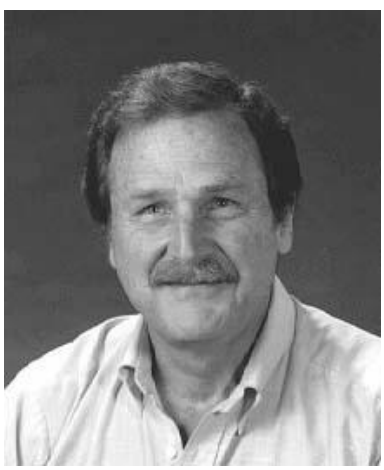




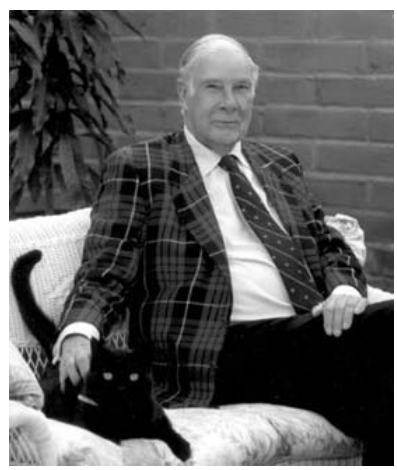
Клод Шеннон



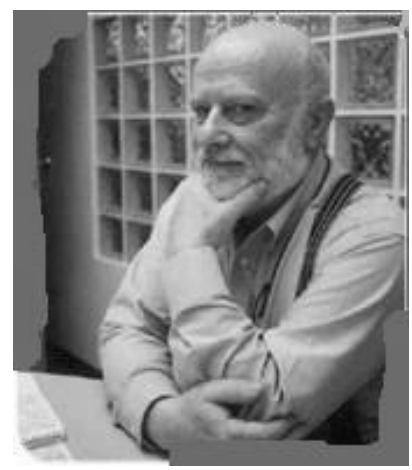
Норберт Винер



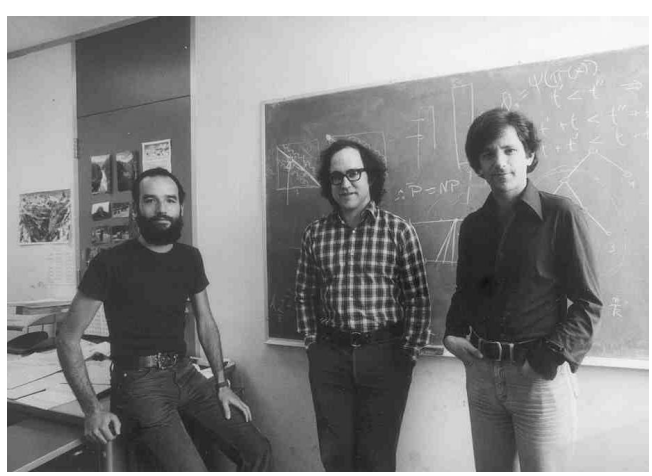
Давид Хаффмен



Ричард Хэмминг



Авраам Лемпел



Рональд Ривест, Эди Шамир, Леонард Адлеман

Дональд Кнут

Достаточность доказывается конструктивно: если условие утверждения выполнено на E, то в качестве декодирующей функции D следует взять функцию, возвращающую ближайшее к декодируемому слово из образа E. Необходимость доказывается от противного. Пусть расстояние между выбранными словами в коде равно $2k$. Тогда если при передаче каждого из этих слов случится k ошибок, которые изменят биты, в которых различаются эти слова, то приемник получит два идентичных сообщения, что свидетельствует о том, что в данной ситуации исправление k ошибок невозможно. Следовательно, минимальное расстояние между словами кода должно быть бóльшим $2k$.

Пример. Рассмотрим $(1,3)$-код, состоящий из E, задающей отображение $0 \to 000$ и $1 \to 111$, и D, задающей отображение $000 \to 0, 001 \to 0, 010 \to 0, 011 \to 1, 100 \to 0, 101 \to 1, 110 \to 1, 111 \to 1$. Этот код (с тройным повторением) исправляет ошибки в одной позиции, т. к. минимальное расстояние между словами кода равно 3.

Если код исправляет все ошибки кратности k и меньшей, то вероятность ошибочного приема слова длины n очевидно не превосходит $\sum_{i=k+1}^{n} C_n^i p^{n-i} q^i$. Вероятность правильного приема в этом случае не меньше, чем

$$\sum_{i=0}^{k} C_n^i p^{n-i} q^i = p^n + C_n^1 p^{n-1} q + \cdots + C_n^k p^{n-k} q^k.$$

Передачу данных часто удобно рассматривать следующим образом. Пусть исходное сообщение $a = a_1 \dots a_m$ кодируется функцией E в кодовое слово $b = b_1 \dots b_n$. Канал связи при передаче добавляет к нему функцией T *строку ошибок* $e = e_1 \dots e_n$ так, что приемник получает сообщение $r = r_1 \dots r_n$, где $r_i = b_i + e_i \, (1 \leqslant i \leqslant n)$. Система, исправляющая ошибки, переводит r в некоторое (обычно ближайшее) кодовое слово. Система, только обнаруживающая ошибки, лишь проверяет, является ли принятое слово кодовым, и сигнализирует о наличии ошибки, если это не так.

Пример. Пусть передаваемое слово $a = 01$ кодируется словом $b = 0110$, а строка ошибок — $e = 0010$. Тогда будет принято слово $r = 0100$. Система, исправляющая ошибки, переведет его в 0110 и затем восстановит переданное слово 01.

Если система только обнаруживает ошибки и расстояние между любыми кодовыми словами $k \geqslant 2$, то любая строка ошибок e с единственной единицей приведет к слову $r = b + e$, которое не является кодовым.

Пример. Рассмотрим $(2,3)$-код с проверкой четности. Множество кодовых слов — $\{000, 011, 101, 110\}$. Ни одна из строк ошибок 001, 010, 100, 111 не переводит одно кодовое слово в другое. Поэтому однократная и тройная ошибки могут быть обнаружены.

Пример. Следующий $(2,5)$-код обнаруживает две ошибки:

$$a_1 = 00 \to 00000 = b_1, \qquad a_2 = 01 \to 01011 = b_2,$$

$$a_3 = 10 \to 10101 = b_3, \qquad a_4 = 11 \to 11110 = b_4.$$

Этот же код способен исправлять однократную ошибку, потому что любые два кодовых слова отличаются по меньшей мере в трех позициях. Из того, что $d(b_i, b_j) \geqslant 3$ при $i \neq j$, следует, что однократная ошибка приведет к приему слова, которое находится на расстоянии 1 от кодового слова, которое было передано. Поэтому схема декодирования, состоящая в том, что принятое слово переводится в ближайшее к нему кодовое, будет исправлять однократную ошибку. В двоичном симметричном канале вероятность правильной передачи одного блока будет не меньше чем $p^5 + 5p^4q$.

Установлено [20], что в $(n-r, n)$-коде, минимальное расстояние между кодовыми словами которого $2k+1$, числа n, r (число дополнительных разрядов в кодовых словах) и k должны соответствовать неравенству

$$r \geqslant \log_2(C_n^k + C_n^{k-1} + \cdots + C_n^1 + 1),$$

называемому *неравенством* или *нижней границей Хэмминга*. Кроме того, если числа n, r и k соответствуют неравенству

$$r > \log_2(C_{n-1}^{2k-1} + C_{n-1}^{2k-2} + \cdots + C_{n-1}^1 + 1),$$

называемому *неравенством* или *верхней границей Варшамова-Гильберта*, то существует $(n-r, n)$-код, исправляющий все ошибки веса k и менее [20].

Нижняя граница задает необходимое условие для помехозащитного кода с заданными характеристиками, т. е. любой такой код должен ему соответствовать, но не всегда можно построить код по подобранным, удовлетворяющим условию характеристикам. Верхняя граница задает достаточное условие для существования помехозащитного кода с заданными характеристиками, т. е. по любым подобранным, удовлетворяющим условию характеристикам можно построить им соответствующий код.

► Упражнение 36

Имеется $(8,9)$-код с проверкой четности. Вычислить вероятность того, что в случае ошибки этот код ее не обнаружит, если вероятность ошибки при передаче каждого бита равна 1%. Вычислить также вероятность ошибочной передачи без использования кода. Сделать аналогичные расчеты для случая, когда вероятность ошибки в десять раз меньше.

▶ **Упражнение 37**
Вычислить минимальную и максимальную оценки количества дополнительных разрядов r для кодовых слов длины n, если требуется, чтобы минимальное расстояние между ними было d. Рассмотреть случаи $n = 32$, $d = 3$ и $n = 23$, $d = 7$.

22. Матричное кодирование

Ранее каждая схема кодирования описывалась таблицами, задающими кодовое слово длины n для каждого исходного слова длины m. Для блоков большой длины этот способ требует большого объема памяти и поэтому непрактичен. Например, для $(16, 33)$-кода потребуется $33 * 2^{16} = 2\,162\,688$ бит.

Гораздо меньшего объема памяти требует *матричное кодирование*. Пусть E матрица размерности $m \times n$, состоящая из элементов e_{ij}, где i — это номер строки, а j — номер столбца. Каждый из элементов матрицы e_{ij} может быть либо 0, либо 1. Кодирование реализуется операцией $b = aE$ или $b_j = a_1e_{1j} + a_2e_{2j} + \cdots + a_me_{mj}$, где кодовые слова рассматриваются как векторы-строки, т.е как матрицы размера $1 \times n$.

Пример. Рассмотрим следующую 3×6-матрицу:

$$E = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

Тогда кодирование задается такими отображениями: $000 \to 000000$, $001 \to 001111$, $010 \to 010011$, $011 \to 011100$, $100 \to 100110$, $101 \to 101001$, $110 \to 110101$, $111 \to 111010$.

Рассмотренный пример показывает преимущества матричного кодирования: достаточно запомнить m кодовых слов вместо 2^m слов. Это общий факт.

Кодирование не должно приписывать одно и то же кодовое слово разным исходным сообщениям. Простой способ добиться этого состоит в том, чтобы m столбцов (в предыдущем примере — первых) матрицы E образовывали единичную матрицу. При умножении любого вектора на единичную матрицу получается этот же самый вектор, следовательно, разным векторам-сообщениям будут соответствовать разные вектора систематического кода.

Матричные коды называют также *линейными кодами*. Для линейных $(n-r, n)$-кодов с минимальным расстоянием Хэмминга d существует *нижняя граница Плоткина* (Plotkin) [14] для минимального количества контрольных разрядов r при $n \geqslant 2d - 1$,

$$r \geqslant 2d - 2 - \log_2 d.$$

▶ **Упражнение 38**
Вычислить минимальную оценку по Плоткину количества дополнительных разрядов r для кодовых слов матричного кода, если требуется, чтобы минимальное расстояние между ними было d. Рассмотреть случаи из предыдущего упражнения.

23. **Групповые коды**

Множество всех двоичных слов $a = a_1 \dots a_m$ длины m образует абелеву (коммутативную) группу относительно поразрядного сложения.

Пусть E — кодирующая $m \times n$-матрица, у которой есть $m \times m$-подматрица с отличным от нуля определителем, например, единичная. Тогда отображение $a \to aE$ переводит группу всех двоичных слов длины m в группу кодовых слов длины n.

Предположим, что $a = a_1 \dots a_m = a' + a''$. Тогда для $b = b_1 \cdots b_n = aE$, $b' = a'E$, $b'' = a''E$, получаем

$$b_j = a_1 e_{1j} + a_2 e_{2j} + \cdots + a_m e_{mj} =$$

$$= (a'_1 + a'_2)e_{1j} + (a'_2 + a''_2)e_{2j} + \cdots + (a'_m + a''_m)e_{mj} = b'_j + b''_j,$$

т. е. $b = b' + b''$. Следовательно, взаимно-однозначное отображение группы двоичных слов длины m при помощи заданной матрицы E сохраняет свойства групповой операции, что означает, что кодовые слова образуют группу.

Блочный код называется *групповым*, если его кодовые слова образуют группу.

Если код является групповым, то наименьшее расстояние между двумя кодовыми словами равно наименьшему весу ненулевого слова.

Это следует из соотношения $d(b_i, b_j) = w(b_i + b_j)$.

В предыдущем примере наименьший вес ненулевого слова равен 3. Следовательно, этот код способен исправлять однократную ошибку или обнаруживать однократную и двойную.

При использовании группового кода незамеченными остаются те и только те ошибки, которые отвечают строкам ошибок, в точности равным кодовым словам.

Такие строки ошибок переводят одно кодовое слово в другое.

Следовательно, вероятность того, что ошибка останется необнаруженной, равна сумме вероятностей всех строк ошибок, равных кодовым словам.

В рассмотренном примере вероятность ошибки равна $4p^3q^3 + 3p^2q^4$.

Рассмотрим задачу оптимизации декодирования группового кода с двоичной матрицей кодирования E. Требуется минимизировать вероятность того, что $D(T(aE)) \neq a$.

Схема декодирования состоит из группы G всех слов, которые могут быть приняты ($\#G = 2^n$). Так как кодовые слова B образуют нормальную (нормальность следует из коммутативности G) подгруппу G, то множеству G можно придать структуру таблицы: будем записывать в одну строку те элементы G, которые являются членами одного смежного класса G по B. Первая строка, соответствующая нулевому слову из G, будет тогда всеми кодовыми словами из B, т. е. $b_0, b_1, \ldots, b_{2^m-1}$. В общем случае, если $g_i \in G$, то строка, содержащая g_i (смежный класс g_iB) имеет вид $b_0 + g_i, b_1 + g_i, \ldots, b_{2^m-1} + g_i$.

Лидером каждого из таких построенных смежных классов называется слово минимального веса.

Каждый элемент g из G однозначно представляется в виде суммы $g_i + b_j$, где $g_i \in G$ — лидер соответствующего смежного класса и $b_j \in B$.

Множество классов смежности группы образуют фактор-группу, которая есть фактор-множество множества G по отношению эквивалентности-принадлежности к одному смежному классу, а это означает, что множества, составляющие это фактор-множество, образуют разбиение G. Отсюда следует, что строки построенной таблицы попарно либо не пересекаются, либо совпадают.

Если в рассматриваемой таблице в первом столбце записать лидеры, то полученная таблица называется *таблицей декодирования*. Она имеет вид:

b_0	b_1	b_2	$\cdots$	b_{2^m-1}
g_1	$g_1 + b_1$	$g_1 + b_2$	$\cdots$	$g_1 + b_{2^m-1}$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
$g_{2^{n-m}-1}$	$g_{2^{n-m}-1} + b_1$	$g_{2^{n-m}-1} + b_2$	$\cdots$	$g_{2^{n-m}-1} + b_{2^m-1}$.

То, что строк будет 2^{n-m} следует из теоремы Лагранжа [1], т. к. 2^{n-m} — это порядок фактор-группы G/B ($\#(G/B) = \#(G)/\#(B)$, $\#B = 2^m$).

Декодирование слова $g = b_j + g_i$ состоит в выборе кодового слова b_j в качестве переданного и последующем применении операции, обратной умножению на E. Такая схема декодирования сможет исправлять ошибки.

Для $(3, 6)$-кода из рассматриваемого примера таблица декодирования будет следующей:

000000	100110	010011	110101	001111	101001	011100	111010
100000	000110	110011	010101	101111	001001	111100	011010
010000	110110	000011	100101	011111	011001	001100	101010
001000	101110	011011	111101	000111	100001	010100	110010
000100	100010	010111	110001	001011	101101	011000	111110
000010	100100	010001	110111	001101	101011	011110	111000
000001	100111	010010	110100	001110	101000	011101	111011
000101	100011	010110	110000	001010	101100	011001	111111.

Первая строка в ней — это строка кодовых слов, а первый столбец — это лидеры.

Чтобы декодировать слово $b_j + e$, следует отыскать его в таблице и выбрать в качестве переданного слово в том же столбце и в первой строке.

Например, если принято слово 110011 (2-я строка, 3-й столбец таблицы), то считается, что было передано слово 010011; аналогично, если принято слово 100101 (3-я строка, 4-й столбец таблицы), переданным считается слово 110101, и т. д.

Групповое кодирование со схемой декодирования посредством лидеров исправляет все ошибки, строки которых совпадают с лидерами. Следовательно, вероятность правильного декодирования переданного по двоичному симметричному каналу кода равна сумме вероятностей всех лидеров, включая нулевой.

В рассмотренной схеме вероятность правильной передачи слова будет $p^6 + 6p^5q + p^4q^2$.

Кодовое слово любого столбца таблицы декодирования является ближайшим кодовым словом ко всем прочим словам данного столбца.

Пусть переданное слово b_i принято как $b_i + e$, $d(b_i, b_i + e) = w(e)$, т. е. это расстояние равно весу соответствующего лидера. Расстояние от $b_i + e$ до любого другого кодового слова b_j равно весу их поразрядной суммы, т. е.

$$d(b_j, b_i + e) = w(b_j + b_i + e) = d(b_j + b_i, e) = d(b_k, e) = w(b_k + e) \geqslant w(e),$$

т. к. e — лидер смежного класса, к которому принадлежат как $b_k + e$, так и $b_i + e$.

Доказано, при схеме декодирования лидерами по полученному слову берется ближайшее к нему кодовое.

▶ Упражнение 39

Для кодирующих матриц $E_1 = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix}$, $E_2 = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$:

1. Построить соответственно $(2, 5)$-код и $(3, 4)$-код.
2. Описать основные характеристики полученных кодов: минимальное расстояние между словами; вероятность необнаружения ошибки; максимальную кратность ошибок, до которой включительно они все исправляются или обнаруживаются.
3. Построить таблицы декодирования.
4. Уточнить характеристики полученных кодов, при использовании их для исправления ошибок, т. е. найти вероятность правильной передачи и описать ошибки, исправляемые этими кодами.
5. Во что будут декодированы слова: 10001, 01110, 10101, 1001, 0110, 1101?

24. **Совершенные и квазисовершенные коды**

Групповой (m, n)-код, исправляющий все ошибки веса, не большего k, и никаких других, называется *совершенным.*

Свойства совершенного кода [1]:

1. Для совершенного (m, n)-кода, исправляющего все ошибки веса, не большего k, выполняется соотношение $\sum_{i=0}^{k} C_n^i = 2^{n-m}$. Верно и обратное утверждение;

2. Совершенный код, исправляющий все ошибки веса, не большего k, в столбцах таблицы декодирования содержит все слова, отстоящие от кодовых на расстоянии, не большем k. Верно и обратное утверждение;

3. Таблица декодирования совершенного кода, исправляющего все ошибки в не более чем k позициях, имеет в качестве лидеров все строки, содержащие не более k единиц. Верно и обратное утверждение.

Совершенный код — это лучший код, обеспечивающий максимум минимального расстояния между кодовыми словами при минимуме длины кодовых слов. Совершенный код легко декодировать: каждому полученному слову однозначно ставится в соответствие ближайшее кодовое. Чисел m, n и k $(1 < k < \frac{n-1}{2})$, удовлетворяющих условию совершенности кода очень мало. Но и при подобранных m, n и k совершенный код можно построить только в исключительных случаях.

Если m, n и k не удовлетворяют условию совершенности, то лучший групповой код, который им соответствует называется *квазисовершенным*, если он исправляет все ошибки кратности, не большей k, и некоторые ошибки кратности $k + 1$. Квазисовершенных кодов также очень мало.

Двоичный блочный (m, n)-код называется *оптимальным*, если он минимизирует вероятность ошибочного декодирования. Совершенный или квазисовершенный код — оптимален. Общий способ построения оптимальных кодов пока неизвестен.

Для любого целого положительного числа r существует совершенный (m, n)-код, исправляющий одну ошибку, называемый *кодом Хэмминга* (Hamming), в котором $m = 2^r - r - 1$ и $n = 2^r - 1$.

Действительно, $\sum_{i=0}^{1} C_n^i = 1 + 2^r - 1 = 2^r = 2^{n-m}$.

Порядок построения кода Хэмминга следующий:

1. Выбираем целое положительное число r. Сообщения будут словами длины $m = 2^r - r - 1$, а кодовые слова — длины $n = 2^r - 1$;

2. В каждом кодовом слове $b = b_1 b_2 \dots b_n$ r бит с индексами-степенями двойки $(2^0, 2^1, \dots, 2^{r-1})$ — являются контрольными, остальные — в естественном порядке — битами сообщения. Например, если $r = 4$, то биты b_1, b_2, b_4, b_8 — контрольные, а $b_3, b_5, b_6, b_7, b_9, b_{10}, b_{11}, b_{12}, b_{13}, b_{14}, b_{15}$ — из исходного сообщения;

3. Строится матрица M из $2^r - 1$ строк и r столбцов. В i-ой строке

стоят цифры двоичного представления числа i. Матрицы для $r = 2$, 3 и 4 таковы:

$$M_{3\times 2} = \begin{bmatrix} 01 \\ 10 \\ 11 \end{bmatrix} \qquad M_{7\times 3} = \begin{bmatrix} 001 \\ 010 \\ 011 \\ 100 \\ 101 \\ 110 \\ 111 \end{bmatrix} \qquad M_{15\times 4} = \begin{bmatrix} 0001 \\ 0010 \\ 0011 \\ 0100 \\ 0101 \\ 0110 \\ 0111 \\ 1000 \\ 1001 \\ 1010 \\ 1011 \\ 1100 \\ 1101 \\ 1110 \\ 1111 \end{bmatrix};$$

4. Записывается система уравнений $bM = 0$, где M — матрица из предыдущего пункта. Система состоит из r уравнений. Например, для $r = 3$:

$$\begin{cases} b_4 + b_5 + b_6 + b_7 = 0 \\ b_2 + b_3 + b_6 + b_7 = 0 \\ b_1 + b_3 + b_5 + b_7 = 0 \end{cases};$$

5. Чтобы закодировать сообщение a, берутся в качестве b_j, j не равно степени двойки, соответствующие биты сообщения и отыскиваются, используя полученную систему уравнений, те b_j, для которых j — степень двойки. В каждое уравнение входит только одно b_j, $j = 2^i$. В выписанной системе b_4 входит в 1-е уравнение, b_2 — во второе и b_1 — в третье. В рассмотренном примере сообщение $a = 0111$ будет закодировано кодовым словом $b = 0001111$.

Декодирование кода Хэмминга проходит по следующей схеме. Пусть принято слово $b + e$, где b — переданное кодовое слово, а e — строка ошибок. Так как $bM = 0$, то $(b + e)M = bM + eM = eM$. Если результат нулевой, как происходит при правильной передаче, считается, что ошибок не было. Если строка ошибок имеет единицу в i-й позиции, то результатом произведения eM будет i-я строка матрицы M или двоичное представление числа i. В этом случае следует изменить символ в i-й позиции слова $b + e$, считая позиции слева, с единицы.

Пример. $(4, 7)$-код Хэмминга имеет в качестве одного из кодовых слов $b = 0001111$. Матрица $M_{7\times 3}$ приведена на шаге 3 хода построения кода Хэмминга. Ясно, что $bM = 0$. Добавим к b строку ошибок $e =$

0010000. Тогда $b + e = 0011111$ и $(b + e)M = 011 = 3_{10}$, т. е. ошибка находится в третьей позиции. Если $e = 0000001$, то $b + e = 0001110$ и позиция ошибки — $(b+e)M = 111 = 7_{10}$ и т. п. Если ошибка допущена в более чем в одной позиции, то декодирование даст неверный результат.

Код Хэмминга — это групповой код.

Это следует из того, что (m, n)-код Хэмминга можно получить матричным кодированием, при помощи $m \times n$-матрицы, в которой столбцы с номерами не степенями 2 образуют единичную подматрицу. Остальные столбцы соответствуют уравнениям шага 4 построения кода Хэмминга, т. е. 1-му столбцу соответствует уравнение для вычисления 1-го контрольного разряда, 2-му — для 2-го, 4-му — для 4-го и т. д. Такая матрица будет при кодировании копировать биты сообщения в позиции не степени 2 кода и заполнять другие позиции кода согласно схеме кодирования Хэмминга.

Пример. Кодирующая матрица для $(4, 7)$-кода Хэмминга —

$$E = \begin{bmatrix} 1110000 \\ 1001100 \\ 0101010 \\ 1101001 \end{bmatrix}.$$

Ее столбцы с номерами 3, 5, 6 и 7 образуют единичную подматрицу. Столбцы с номерами 1, 2 и 4 соответствуют уравнениям для вычисления контрольных бит, например, уравнению $b_1 = b_3 + b_5 + b_7$ соответствует столбец 1101, т. е. для вычисления первого контрольного разряда берутся 1, 2 и 4 биты исходного сообщения или биты 3, 5 и 7 кода.

К (m, n)-коду Хэмминга можно добавить проверку четности. Получится $(m, n + 1)$-код с наименьшим весом ненулевого кодового слова 4, способный исправлять одну и обнаруживать две ошибки.

Коды Хэмминга накладывают ограничения на длину слов сообщения: эта длина может быть только числами вида $2^r - r - 1$: 1, 4, 11, 26, 57, ... Но в реальных системах информация передается байтам или машинными словами, т. е. порциями по 8, 16, 32 или 64 бита, что делает использование совершенных кодов не всегда подходящим. Поэтому в таких случаях часто используются квазисовершенные (m, n)-коды.

Квазисовершенные (m, n)-коды, исправляющие одну ошибку, строятся следующим образом. Выбирается минимальное n так, чтобы

$$\frac{2^n}{n+1} \geqslant 2^m.$$

Каждое кодовое слово такого кода будет содержать $k = n - m$ контрольных разрядов. Из предыдущих соотношений следует, что

$$2^k = 2^{n-m} \geqslant n + 1 = C_n^1 + C_n^0 = m + k + 1.$$

Каждому из n разрядов присваивается слева-направо номер от 1 до n. Для заданного слова сообщения составляются k контрольных сумм $S_1, \ldots, S_k$ по модулю 2 значений специально выбранных разрядов кодового слова, которые помещаются в позиции-степени 2 в нем: для S_i $(1 \leqslant i \leqslant k)$ выбираются разряды, содержащие биты исходного сообщения, двоичные числа-номера которых имеют в i-м разряде единицу. Для суммы S_1 это будут, например, разряды 3, 5, 7 и т. д., для суммы S_2 — 3, 6, 7 и т. д. Таким образом, для слова сообщения $a = a_1 \ldots a_m$ будет построено кодовое слово $b = S_1 S_2 a_1 S_3 a_2 a_3 a_4 S_4 a_5 \ldots a_m$. Обозначим S_i^* сумму по модулю 2 разрядов полученного слова, соответствующих контрольной сумме S_i и самой этой контрольной суммы. Если $S_k^* \ldots S_1^* = 0$, то считается, что передача прошла без ошибок. В случае одинарной ошибки $S_k^* \ldots S_1^*$ будет равно двоичному числу-номеру сбойного бита. В случае ошибки, кратности большей 1, когда $S_k^* \ldots S_1^* > n$, ее можно обнаружить. Подобная схема декодирования не позволяет исправлять некоторые двойные ошибки, чего можно было бы достичь, используя схему декодирования с лидерами, но последняя значительно сложнее в реализации и дает незначительное улучшение качества кода.

Пример построения кодового слова квазисовершенного $(9, n)$-кода, исправляющего все однократные ошибки, для сообщения 100011010.

$$\frac{2^{12}}{13} = \frac{4096}{13} < 2^9 = 512 \quad \text{и} \quad \frac{2^{13}}{14} = \frac{4096}{7} > 512, \text{ т. е. } n = 13.$$

Искомое кодовое слово имеет вид $\overset{1}{S_1}\overset{2}{S_2}\ 1\ \overset{4}{S_3}\ 0\ \ 0\ \ 0\ \overset{8}{S_4}\ 1\ \ 1\ \ 0\ \ 1\ \ 0$. Далее нужно вычислить контрольные суммы.

$$\begin{aligned}
1_{10} &= 0000_2 \\
2_{10} &= 0010_2 \\
3_{10} &= 0011_2 \\
4_{10} &= 0100_2 \\
5_{10} &= 0101_2 \\
6_{10} &= 0110_2 \\
7_{10} &= 0111_2 \\
8_{10} &= 1000_2 \\
9_{10} &= 1001_2 \\
10_{10} &= 1010_2 \\
11_{10} &= 1011_2 \\
12_{10} &= 1100_2 \\
13_{10} &= 1101_2
\end{aligned}$$

$$\begin{aligned}
S_1 &= b_3 + b_5 + b_7 + b_9 + b_{11} + b_{13} = 0 \\
S_2 &= b_3 + b_6 + b_7 + b_{10} + b_{11} = 0 \\
S_3 &= b_5 + b_6 + b_7 + b_{12} + b_{13} = 1 \\
S_4 &= b_9 + b_{10} + b_{11} + b_{12} + b_{13} = 1
\end{aligned}$$

Таким образом, искомый код — 0011000111010. Если в процессе передачи этого кода будет испорчен его пятый бит, то приемник получит код

0011100111010. Для его декодирования опять вычисляются контрольные суммы:

$$\begin{aligned} S_1^* &= b_1 + b_3 + b_5 + b_7 + b_9 + b_{11} + b_{13} = 1 \\ S_2^* &= b_2 + b_3 + b_6 + b_7 + b_{10} + b_{11} = 0 \\ S_3^* &= b_4 + b_5 + b_6 + b_7 + b_{12} + b_{13} = 1 \\ S_4^* &= b_8 + b_9 + b_{10} + b_{11} + b_{12} + b_{13} = 0 \end{aligned}$$

$$S_4^* S_3^* S_2^* S_1^* = 0101_2 = 5_{10}.$$

Приемник преобразует изменением пятого бита полученное сообщение в отправленное передатчиком, из которого затем отбрасыванием контрольных разрядов восстанавливает исходное сообщение.

Совершенный код Хэмминга также можно строить по рассмотренной схеме, т. к. для него $2^n/(n+1) = 2^m$.

Для исправление одинарной ошибки к 8-разрядному коду достаточно приписать 4 разряда ($2^{12}/13 > 2^8$), к 16-разрядному — 5, к 32-разрядному — 6, к 64-разрядному — 7.

▶ Упражнение 40
Может ли $(6, 14)$-код, минимальное расстояние между кодовыми словами которого 5, быть совершенным?

▶ Упражнение 41
Построить кодовые слова квазисовершенного $(9, n)$-кода, исправляющего однократные ошибки, для тех сообщений, которые соответствуют числам 55, 200 и декодировать слова 1000001000001, 1100010111100, полученные по каналу связи, использующему этот код.

25. **Полиномиальные коды**

При *полиномиальном кодировании* каждое сообщение отождествляется с многочленом, а само кодирование состоит в умножении на фиксированный многочлен. Полиномиальные коды — блочные и отличаются от рассмотренных ранее только алгоритмами кодирования и декодирования.

Пусть $a = a_0 \dots a_{m-1}$ — двоичное сообщение. Тогда сопоставим ему многочлен $a(x) = a_0 + a_1 x + \dots + a_{m-1} x^{m-1}$. Все вычисления происходят в поле классов вычетов по модулю 2, т. е. от результата любой арифметической операции берется остаток от его деления на 2.

Например, последовательности 10011 при $m = 5$ соответствует многочлен $1 + x^3 + x^4$.

Зафиксируем некоторый многочлен степени k,

$$g(x) = g_0 + g_1 x + \dots + g_k x^k, \quad g_0 \neq 0, \quad g_k \neq 0.$$

Полиномиальный код с кодирующим многочленом $g(x)$ кодирует слово сообщения $a(x)$ многочленом $b(x) = a(x)g(x) = b_0 + b_1x + \cdots + b_{n-1}x^{n-1}$ или кодовым словом из коэффициентов этого многочлена $b = b_0 \dots b_{n-1}$. Условия $g_0 \neq 0$ и $g_k \neq 0$ необходимы, потому что в противном случае b_0 и b_{n-1} не будут нести никакой информации, т. к. они всегда будут нулями.

Пример. Рассмотрим кодирующий многочлен $g(x) = 1 + x^2 + x^3$. Сообщение 01011, отвечающее многочлену $a(x) = x + x^3 + x^4$, будет закодировано коэффициентами многочлена $b(x) = g(x)a(x) = x+x^5+x^7$, т. е. $b = 01000101$.

Полиномиальный код с кодирующим многочленом $g(x)$ степени k является матричным кодом с кодирующей матрицей G размерности $m\times(m+k)$:

$$G = \begin{bmatrix} g_0 & g_1 & g_2 & \cdots & g_k & 0 & 0 & \cdots & 0 \\ 0 & g_0 & g_1 & \cdots & g_{k-1} & g_k & 0 & \cdots & 0 \\ 0 & 0 & g_0 & \cdots & g_{k-2} & g_{k-1} & g_k & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & g_k \end{bmatrix}.$$

Т. е. ненулевые элементы в j-й строке — это последовательность коэффициентов кодирующего многочлена, расположенных с j-го по $(j+k)$-й столбцах.

Например, $(3, 6)$-код с кодирующим многочленом $1+x+x^3$ отвечает матрице

$$G = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$

или отображению: $000 \to 000000$; $001 \to 001101$; $010 \to 011010$; $011 \to 010111$; $100 \to 110100$; $101 \to 111001$; $110 \to 101110$; $111 \to 100011$.

Полиномиальные коды являются групповыми.

Это следует из того, что коды, получаемые матричным кодированием, — групповые.

Рассмотрим (m, n)-код с кодирующим многочленом $g(x)$. Строка ошибок $e = e_0 \dots e_{n-1}$ останется необнаруженной в том и только в том случае, если соответствующий ей многочлен $e(x) = e_0 + e_1x + \cdots + e_{n-1}x^{n-1}$ делится на $g(x)$.

Действительно, $a(x)g(x) + e(x)$ делится на $g(x)$ тогда и только тогда, когда $e(x)$ делится на $g(x)$. Поэтому любая ошибка, многочлен которой не делится на $g(x)$, будет обнаружена и, соответственно, любая ошибка, многочлен которой делится на $g(x)$, не может быть обнаружена.

Таким образом, обнаружение ошибки при использовании полиномиального кода с кодирующим многочленом $g(x)$ может быть реализовано при помощи алгоритма деления многочленов с остатком: если остаток ненулевой, то при передаче произошло искажение данных.

Коды Хэмминга можно строить как полиномиальные, например, кодирующий многочлен $x^3 + x^2 + 1$ определяет совершенный $(4, 7)$-код, отличный от рассмотренного ранее.

Вообще же, если кодирующий многочлен $g(x)$, порождающий соответствующий (m, n)-код, не является делителем ни одного из многочленов вида $x^j + 1$ при $j < n$, то минимальное расстояние между кодовыми словами порожденного им кода не меньше 3.

Пусть d — минимальное расстояние между кодовыми словами, оно равно минимуму среди весов ненулевых кодовых слов. Предположим $d = 2$. Тогда существует $a(x)$ такой, что $a(x)g(x) = b(x)$ и степень $b(x)$ не больше n. Вес b равен 2, поэтому $b(x) = x^m + x^l$ и $l < m < n$. Следовательно, $b(x) = x^l(x^{m-l} + 1)$, что означает, что $x^{m-l} + 1$ должен делиться на $g(x)$, а это невозможно по условию. Если предположить, что $d = 1$, то это приведет к утверждению о том, что x^m должен делиться на $g(x)$, что тоже противоречит условию. Итак, $d \geqslant 3$.

Кодирующий многочлен $x^{11} + x^9 + x^7 + x^6 + x^5 + x + 1$ определяет совершенный $(12, 23)$-код Голея (Golay) с минимальным расстоянием между кодовыми словами 7.

В 1971 году финскими и советскими математиками было доказано [20], что кроме кодов Хэмминга и Голея других совершенных кодов нет.

Наиболее интересными среди полиномиальных кодов являются *циклические коды*, в которых вместе с любым кодовым словом вида $b_0 \dots b_{n-2} b_{n-1}$ есть кодовое слово $b_{n-1} b_0 \dots b_{n-2}$.

▶ Упражнение 42
По кодирующему многочлену $x^7 + x^5 + x + 1$ построить полиномиальные коды для двоичных сообщений 0100, 10001101, 11110.

▶ Упражнение 43
Принадлежат ли коду Голея кодовые слова 100001010111111010011111 и 110001110111100100111111?

26. Понятие о кодах Боуза-Чоудхури-Хоккенгема

Остался открытым вопрос о методике построения кодов, минимальное расстояние между кодовыми словами которых равно заданному числу. В 1960 году независимо Боуз (Bose), Чоудхури (Chaudhuri) и Хоккенгем (Hocquengem) открыли способ построения полиномиальных кодов, удовлетворяющих таким требованиям. Эти коды получили названия кодов Боуза-Чоудхури-Хоккенгема или БЧХ-кодов (BCH codes).

БЧХ-коды могут быть не только двоичными, например, на практике достаточно широко используются недвоичные коды Рида-Соломона (Reed, Solomon), но далее будут рассматриваться только двоичные.

Многочлен $g(x)$ степени k называется *примитивным*, если $x^j + 1$ делится на $g(x)$ без остатка для $j = 2^k - 1$ и не делится ни для какого меньшего значения j.

Например, многочлен $1 + x^2 + x^3$ примитивен: он делит $x^7 + 1$, но не делит $x^j + 1$ при $j < 7$. Примитивен также многочлен $1 + x^3 + x^4$ — он делит $x^{15} + 1$, но не делит $x^j + 1$ при $j < 15$.

Кодирующий многочлен $g(x)$ для БЧХ-кода, длина кодовых слов которого n, строится так. Находится примитивный многочлен минимальной степени q такой, что $n \leqslant 2^q - 1$ или $q \geqslant \log_2(n + 1)$. Пусть α — корень этого многочлена, тогда рассмотрим кодирующий многочлен $g(x) = \text{НОК}(m_1(x), \ldots, m_{d-1}(x))$, где $m_1(x), \ldots, m_{d-1}(x)$ — многочлены минимальной степени, имеющие корнями соответственно $\alpha, \alpha^2, \ldots, \alpha^{d-1}$.

Построенный кодирующий многочлен производит код с минимальным расстоянием между кодовыми словами, не меньшим d, и длиной кодовых слов n [1].

Пример. Нужно построить БЧХ-код с длиной кодовых слов $n = 15$ и минимальным расстоянием между кодовыми словами $d = 5$. Степень примитивного многочлена равна $q = \log_2(n + 1) = 4$ и сам он равен $x^4 + x^3 + 1$. Пусть α — его корень, тогда α^2 и α^4 — также его корни. Минимальным многочленом для α^3 будет $x^4 + x^3 + x^2 + x + 1$. Следовательно,

$$g(x) = \text{НОК}(x^4 + x^3 + 1, x^4 + x^3 + x^2 + x + 1) =$$

$$= (x^4 + x^3 + 1)(x^4 + x^3 + x^2 + x + 1) = x^8 + x^4 + x^2 + x + 1.$$

Степень полученного многочлена равна 8, следовательно, построенный БЧХ-код будет $(7, 15)$-кодом. Слово 1000100 или $a(x) = x^4 + 1$ будет закодировано кодовым словом $a(x)g(x) = x^{12} + x^6 + x^5 + x^2 + x + 1$ или 111001100000100.

Можно построить [1] двоичный БЧХ-код с кодовыми словами длины $n = 2^q - 1$ и нечетным минимальным расстоянием d, у которого число контрольных символов не больше $\dfrac{q(d-1)}{2}$.

Первый БЧХ-код, примененный на практике, был $(92, 127)$-кодом, исправляющим ошибки кратности до 5, но наиболее широкое распространение получил $(231, 255)$-код, обнаруживающий ошибки кратности до 6.

БЧХ-коды умеренной длины не слишком далеки от совершенных или квазисовершенных кодов. Коды Хэмминга, например, являются БЧХ-кодами, а БЧХ-коды с минимальным весом кодового слова 5 —

квазисовершенны. Но с ростом длины кодовых слов качество БЧХ-кодов падает. Код Голея, например, — это не код БЧХ.

▶ Упражнение 44
Найти кодирующий многочлен БЧХ-кода $g(x)$ с длиной кодовых слов 15 и минимальным расстоянием между кодовыми словами 7. Использовать примитивный многочлен $m_1(x) = 1 + x + x^4$ с корнем α. Проверить, будут ли α^3 и α^5 корнями соответственно многочленов $m_3(x) = 1 + x + x^2 + x^3 + x^4$ и $m_5(x) = 1 + x + x^2$.

27. **Циклические избыточные коды**

Циклический избыточный код (Cyclical Redundancy Check — CRC) имеет фиксированную длину и используется для обнаружения ошибок. Наибольшее распространения получили коды CRC-16 и CRC-32, имеющие длину 16 и 32 бита соответственно. Код CRC строится по исходному сообщению произвольной длины, т. е. этот код не является блочным в строгом смысле этого слова. Но при каждом конкретном применении этот код — блочный, $(m, m + 16)$-код для CRC-16 или $(m, m + 32)$-код для CRC-32.

Вычисление значения кода CRC происходит посредством деления многочлена, соответствующего исходному сообщению (полином-сообщение), на фиксированный многочлен (полином-генератор). Остаток от такого деления и есть код CRC, соответствующий исходному сообщению. Для кода CRC-16 полином-генератор имеет степень 16, а для CRC-32 — 32. Полиномы-генераторы подбираются специальным образом и для кодов CRC-16/32 стандартизированы Международным консультативным комитетом по телеграфной и телефонной связи (CCITT). Для CRC-16, например, стандартным является полином-генератор $x^{16} + x^{12} + x^5 + 1$.

Пример построения CRC-4 кода для сообщения 11010111, используя полином-генератор $x^4+x^3+x^2+1$. Исходному сообщению соответствует полином $x^7+x^6+x^4+x^2+x+1$, т. е. нумерация битов здесь начинается справа.

$$\begin{array}{l|l} x^7 + x^6 + x^4 + x^2 + x + 1 & x^4 + x^3 + x^2 + 1 \\ \underline{x^7 + x^6 + x^5 + x^3} & \hline x^3 + x \\ x^5 + x^4 + x^3 + x^2 + x + 1 & \\ \underline{x^5 + x^4 + x^3 + x} & \\ x^2 + 1 & \end{array}$$

Полиному $x^2 + 1$ соответствуют биты 0101 — это и есть CRC-4 код.

Существуют быстрые алгоритмы для расчета CRC-кодов, использующие специальные таблицы, а не деление многочленов с остатком.

CRC-коды способны обнаруживать одиночную ошибку в любой позиции и, кроме того, многочисленные комбинации кратных ошибок, рас-

положенных близко друг от друга. При реальной передаче или хранении информации ошибки обычно группируются на некотором участке, а не распределяются равномерно по всей длине данных. Таким образом, хотя для идеального случая двоичного симметричного канала CRC-коды не имеют никаких теоретических преимуществ по сравнению, например, с простыми контрольными суммами, для реальных систем эти коды являются очень полезными.

Коды CRC используются очень широко: модемами, телекоммуникационными программами, программами архивации и проверки целостности данных и многими другими программными и аппаратными компонентами вычислительных систем.

▶ Упражнение 45
Построить CRC-4 код для сообщений 10000000 и 101111001, используя полином-генератор $x^4 + 1$.

28. **Основы теории защиты информации**

Криптография (тайнопись) — это раздел математики, в котором изучаются и разрабатываются системы изменения письма с целью сделать его непонятным для непосвященных лиц. Известно, что еще в V веке до нашей эры тайнопись использовалась в Греции. В современном мире, где все больше и больше услуг предоставляется через использование информационных технологий, проблема защиты информации методами криптографии имеет первостепенное значение. Сегодня большая часть обмена информацией проходит по компьютерным сетям и часто (в бизнесе, военным и прочее) нужно обеспечивать конфиденциальность такого обмена. Теоретические основы классической криптографии впервые были изложены Клодом Шенноном в конце 1940-х годов.

Простейшая система шифрования — это замена каждого знака письма на другой знак по выбранному правилу. Юлий Цезарь, например, заменял в своих секретных письмах первую букву алфавита на четвертую, вторую — на пятую, последнюю — на третью и т. п., т. е. A на D, B на E, Z на C и т. п. Подобные шифры, называемые простой заменой или подстановкой, описаны в рассказах “Пляшущие человечки” А. К. Дойла, “Золотой жук” Э. По и других.

Шифры *простой замены* легко поддаются расшифровке, при знании исходного языка сообщения, т. к. каждый письменный язык характеризуется частотой встречаемости своих знаков. Например, в английском языке чаще всего встречается буква E, а в русском — О. Таким образом, в шифрованном подстановкой сообщении на русском языке самому частому знаку будет с большой вероятностью соответствовать буква О. Вероятность будет расти с ростом длины сообщения.

Усовершенствованные шифры-подстановки используют возможность заменять символ исходного сообщения на любой символ из заданного для него множества символов, что позволяет выровнять частоты встречаемости различных знаков шифра, но подобные шифры удлиняют сообщение и замедляют скорость обмена информацией.

В шифрах-*перестановках* знаки сообщения специальным образом переставляются между собой, например, записывая сообщение в строки заданной длины и беря затем последовательность слов в столбцах в качестве шифра. Сообщение “ТЕОРИЯИНФОРМАЦИИ”, используя строки длины 4, будет в шифрованном таким методом виде выглядеть как “ТИФАЕЯОЦОИРИРНМИ”, потому что при шифровании использовался следующий прямоугольник:

```
ТЕОР
ИЯИН
ФОРМ
АЦИИ.
```

Шифры-перестановки в общем случае практически не поддаются дешифровке. Для их дешифровки необходимо знать дополнительную информацию. Крупный недостаток подобных шифров в том, что если удастся каким-то образом расшифровать хотя бы одно сообщение, то в дальнейшем можно расшифровать и любое другое. Модификацией шифров-перестановок являются шифры-перестановки со словом-ключом, которое определяет порядок взятия слов-столбцов. Например, если для рассмотренного шифра взять ключ “РЫБА”, то шифрованное сообщение будет выглядеть как “РНМИОИРИТИФАЕЯОЦ”.

Системы с ключевым словом или просто ключом, известные с XVI века, широко применяются до сих пор. Их особенностью является два уровня секретности. Первый уровень — это собственно способ составления кода, который постоянно известен лицам, использующим данный шифр. Второй уровень — это ключ, который посылается отдельно от основного сообщения по особо защищенным каналам и без которого расшифровка основного сообщения невозможна.

Наиболее простой способ использования ключа хорошего шифра следующий: под символами сообщения записывается раз за разом ключ, затем номера соответствующих знаков сообщения и ключа складываются. Если полученная сумма больше общего числа знаков, то от нее отнимается это общее число знаков. Полученные числа будут номерами символов кода. С ростом длины ключа трудоемкость дешифровки подобного шифра стремительно растет. Например, рассмотренное ранее сообщение с ключом “КИБЕРНЕТИКА” в шифрованном виде будет выглядеть как “ЮОРЦЪНОБЮЪСШЙШОЪ”. Процесс шифровки описывается схемой:

Т	Е	О	Р	И	Я	И	Н	Ф	О	Р	М	А	Ц	И	И
20	6	16	18	10	33	10	15	22	16	18	14	1	24	10	10
К	И	Б	Е	Р	Н	Е	Т	И	К	А	К	И	Б	Е	Р
12	10	2	6	18	15	6	20	10	12	1	12	10	2	6	18
32	16	18	24	28	15	16	2	32	28	19	26	11	26	16	28
Ю	О	Р	Ц	Ъ	Н	О	Б	Ю	Ъ	С	Ш	Й	Ш	О	Ъ.

Если в качестве ключа использовать случайную последовательность, то получится *нераскрываемый шифр*. Проблема этого шифра — это способ передачи ключа.

В информационных сетях использование традиционных систем шифрования с ключом затрудненно необходимостью иметь специальный особо защищенный способ для передачи ключа. В 1976 году У. Диффи (Diffie W.) и М. Хеллман (Hellman M.) — инженеры-электрики из Станфордского университета, а также студент Калифорнийского университета Р. Меркль (Merkle R.), предложили новый принцип построения криптосистем, не требующий передачи ключа принимающему сообщение и сохранения в тайне метода шифрования. В дальнейшем, в качестве примеров, рассмотрим три системы, основанные на идеях Диффи и Хеллмана: без передачи ключей, с открытым ключом и электронную подпись — все они в свою очередь основаны на математическом фундаменте теории чисел.

► Упражнение 46
Зашифровать сообщение “КИБЕРНЕТИКА” ключом “ДИСК”.

29. **Криптосистема без передачи ключей**

Пусть абоненты $A, B, C, \ldots$ условились организовать между собой секретную переписку. Для этой цели они выбирают достаточно большое простое число p такое, что $p-1$ хорошо разлагается на не очень большие простые множители. Затем каждый из абонентов независимо один от другого выбирает себе некоторое натуральное число, взаимно простое с $p-1$. Пусть число абонента A — a, абонента B — b и т. д. Числа $a, b, \ldots$ составляют первые секретные ключи соответствующих абонентов. Вторые секретные ключи (α для A, β для B и т. д.) находятся из уравнений: для A из $a\alpha \equiv 1 \pmod{\varphi(p)}$, $0 < \alpha < p-1$; для B — из $b\beta \equiv 1 \pmod{\varphi(p)}$, $0 < \beta < p-1$ и т. д. Пересылаемые сообщения, коды-числа, должны быть меньше $p-1$. В случае, когда сообщение больше или равно $p-1$, оно разбивается на части таким образом, чтобы каждая часть была числом, меньшим $p-1$.

Предположим абонент A решил отправить сообщение m ($m < p-1$) B. Для этого он сначала зашифровывает свое сообщение ключом a, получая по формуле $m_1 \equiv m^a \pmod{p}$ шифрованное сообщение m_1, которое отправляется B. B, получив m_1, зашифровывает его своим ключом

b, получая по формуле $m_2 \equiv m_1^b \pmod p$ шифрованное сообщение m_2, которое отправляется обратно к A. A шифрует полученное сообщение ключом α по формуле $m_3 \equiv m_2^\alpha \pmod p$ и окончательно отправляет m_3 к B. B, используя ключ β, сможет теперь расшифровать исходное сообщение m. Действительно, $m_4 \equiv m_3^\beta \equiv m^{a\alpha b\beta} \equiv m \pmod p$, т. к. $a\alpha b\beta \equiv 1 \pmod{\varphi(p)}$, следовательно, $a\alpha b\beta = k\varphi(p) + 1$ для некоторого целого k и $m^{k\varphi(p)+1} \equiv (m^{\varphi(p)})^k m \equiv m \pmod p$, т. к. $m^{\varphi(p)} \equiv 1 \pmod p$ по теореме Эйлера-Ферма.

Пример. Абоненты A и B вместе выбрали $p = 23$ ($\varphi(23) = 22$), A выбрал $a = 5$, а B — $b = 7$. Затем из уравнения $5\alpha \equiv 1 \pmod{\varphi(23)}$ A находит $\alpha = 9$, а B из подобного уравнения находит $\beta = 19$. При передаче сообщения $m = 17$ от A к B сначала A отправляет к B $m_1 \equiv 17^5 \equiv 21 \pmod{23}$, из $m_1 = 21$ B вычисляет $m_2 \equiv 21^7 \equiv 10 \pmod{23}$ и отправляет его обратно A, из $m_2 = 10$ A вычисляет для B $m_3 \equiv 10^9 \equiv 20 \pmod{23}$, наконец, B может прочитать посланное ему сообщение $20^{19} \equiv 17 \pmod{23}$.

▶ Упражнение 47
Между абонентами A и B установлен секретный канал связи без передачи ключей при заданных $p = 167$ и их первых ключах 15 и 21. Описать процесс передачи сообщений 22 (от A к B) и 17 (от B к A).

30. **Криптосистема с открытым ключом**

Первую и наиболее известную систему с открытым ключом разработали в 1978 году американцы Р. Ривест (Rivest R.), Э. Шамир (Shamir A.) и Л. Адлеман (Adleman L.). По их именам эта система получила название RSA.

Пусть абоненты A и B решили организовать для себя возможность секретной переписки. Для этого каждый из них независимо выбирает два больших простых числа (p_{A_1}, p_{A_2} и p_{B_1}, p_{B_2}), находит их произведение (r_A и r_B), функцию Эйлера от этого произведения ($\varphi(r_A)$ и $\varphi(r_B)$) и случайное число (a и b), меньшее вычисленного значения функции Эйлера и взаимно простое с ним. Кроме того, A из уравнения $a\alpha \equiv 1 \pmod{\varphi(r_A)}$ находит α ($0 < \alpha < \varphi(r_A)$), а B из уравнения $b\beta \equiv 1 \pmod{\varphi(r_B)}$ находит β ($0 < \beta < \varphi(r_B)$). Затем A и B печатают доступную всем книгу паролей вида:

A: r_A, a
B: r_B, b

.

Теперь кто-угодно может отправлять конфиденциальные сообщения A или B. Например, если пользователь книги паролей хочет отправить сообщение m для B (m должно быть меньшим r_B, или делиться на куски, меньшие r_B), то он использует ключ b из книги паролей для получения шифрованного сообщения m_1 по формуле $m_1 \equiv m^b$

$(\mathrm{mod}\ r_B)$, которое и отправляется B. B для дешифровки m_1 использует ключ β в формуле $m_1^{\beta} \equiv m^{b\beta} \equiv m \pmod{r_B}$, т. к. $b\beta \equiv 1 \pmod{\varphi(r_B)}$, следовательно, $b\beta = k\varphi(r_B) + 1$ для некоторого целого k и $m^{k\varphi(r_B)+1} \equiv (m^{\varphi(r_B)})^k m \equiv m \pmod{r_B}$, т.к. $m^{\varphi(r_B)} \equiv 1 \pmod{r_B}$ по теореме Эйлера-Ферма. Доказано [12], что задача нахождения секретного ключа β по данным из книги паролей имеет ту же сложность, что и задача разложения числа r_B на простые множители.

Пример. Пусть для A $p_{A_1} = 7$ и $p_{A_2} = 23$, тогда $r_A = p_{A_1} p_{A_2} = 161$, $\varphi(161) = 6 * 22 = 132$, $a = 7$, $\alpha = 19$ (из уравнения $7\alpha \equiv 1 \pmod{132}$). Следовательно, запись в книге паролей для A будет иметь вид A: $161, 7$. Если кто-то захочет отправить A секретное сообщение $m = 3$, то он должен сначала превратить его в шифровку m_1 по формуле $m_1 \equiv 3^7 \equiv 94 \pmod{161}$. Когда A получит $m_1 = 94$ он дешифрует его по формуле $m \equiv 94^{19} \equiv 3 \pmod{161}$.

▶ Упражнение 48
Нужно послать секретные сообщения 25 и 2 для JB и 14 для CIA, используя следующие записи открытой книги паролей криптосистемы RSA:

JB: 77,7;
CIA: 667,15.

▶ Упражнение 49
Пользователь системы RSA выбрал $p_1 = 11$ и $p_2 = 47$. Какие из чисел 12, 33, 125, 513 он может выбрать для открытого ключа? Вычислить для них закрытый ключ.

▶ Упражнение 50
Пользователь системы RSA, выбравший $p_1 = 17$, $p_2 = 11$ и $a = 61$, получил шифрованное сообщение $m_1 = 3$. Дешифровать m_1.

31. Электронная подпись

Криптосистема с открытым ключом открыта для посылки сообщений для абонентов из книги паролей для любого желающего. В системе с электронной подписью сообщение необходимо "подписывать", т.е. явно указывать на отправителя из книги паролей.

Пусть $W_1, W_2, \ldots, W_n$ — абоненты системы с электронной подписью. Все они независимо друг от друга выбирают и вычисляют ряд чисел точно так же как и в системе с открытым ключом. Пусть i-ый абонент $(1 \neq i \leqslant n)$ выбирает два больших простых числа p_{i1} и p_{i2}, затем вычисляет их произведение — $r_i = p_{i1} p_{i2}$ и функцию Эйлера от него — $\varphi(r_i)$, затем выбирает первый ключ a_i из условий $0 < a_i < \varphi(r_i)$, НОД$(a_i, \varphi(r_i)) = 1$ и, наконец, вычисляет второй ключ α_i из уравнения $a_i \alpha_i \equiv 1 \pmod{\varphi(r_i)}$. Записи в книге паролей будут иметь вид:

W_1: r_1, a_1
W_2: r_2, a_2
$\ldots$
W_n: r_n, a_n

.

Если абонент W_1 решает отправить секретное письмо m W_2, то ему следует проделать следующую последовательность операций:

1) Если $m > \min(r_1, r_2)$, то m разбивается на части, каждая из которых меньше меньшего из чисел r_1 и r_2;
2) Если $r_1 < r_2$, то сообщение m сначала шифруется ключом α_1 ($m_1 \equiv m^{\alpha_1} \pmod{r_1}$), а затем — ключом a_2 ($m_2 \equiv m_1^{a_2} \pmod{r_2}$), если же $r_1 > r_2$, то сообщение m сначала шифруется ключом a_2 ($m_1 \equiv m^{a_2} \pmod{r_2}$), а затем — ключом α_1 ($m_2 \equiv m_1^{\alpha_1} \pmod{r_1}$);
3) Шифрованное сообщение m_2 отправляется W_2.

W_2 для дешифровки сообщения m_2 должен знать, кто его отправил, поэтому к m_2 должна быть добавлена электронная подпись, указывающая на W_1. Если $r_1 < r_2$, то для расшифровки m_2 сначала применяется ключ α_2, а затем — a_1, если же $r_1 > r_2$, то для расшифровки m_2 сначала применяется ключ a_1, а затем — α_2. Рассмотрим случай $r_1 < r_2$: $m_2^{\alpha_2} \equiv m_1^{a_2\alpha_2} \equiv m_1 \pmod{r_2}$ и $m_1^{a_1} \equiv m^{\alpha_1 a_1} \equiv m \pmod{r_1}$ по теореме Эйлера-Ферма.

Пример. Пусть W_1 выбрал и вычислил следующие числа $p_{11} = 7, p_{12} = 13, r_1 = p_{11}p_{12} = 91, \varphi(91) = 72, a_1 = 5, \alpha_1 = 29$, а W_2 — следующие $p_{21} = 11, p_{22} = 23, r_2 = 253, \varphi(253) = 220, a_2 = 31, \alpha_2 = 71$. После занесения записей о W_1 и W_2 в открытую книгу паролей, W_2 решает послать сообщение $m = 41$ для W_1. Т.к. $r_2 > r_1$, то сообщение сначала шифруется ключом a_1, а затем ключом α_2: $m_1 \equiv 41^5 \equiv 6 \pmod{91}$, $m_2 \equiv 6^{71} \equiv 94 \pmod{253}$. Сообщение m_2 отправляется W_1. Получив $m_2 = 94$, W_1, зная, что оно пришло от W_2, дешифрует его сначала ключом a_2, а затем ключом α_1: $94^{31} \pmod{253} \equiv 6$, $6^{29} \pmod{91} \equiv 41$.

Если подписать сообщение открытым образом, например, именем отправителя, то такая “подпись” будет ничем не защищена от подделки. Защита электронной подписи обычно реализуется с использованием таких же методов, что в криптосистеме с открытым ключом.

Электронная подпись генерируется отправителем по передаваемому сообщению и секретному ключу. Получатель сообщения может проверить его аутентичность по прилагаемой к нему электронной подписи и открытому ключу отправителя.

Стандартные системы электронной подписи считаются настолько надежными, что электронная подпись юридически приравнена к рукописной. Электронная подпись часто используется с открытыми, незашифрованными электронными документами.

32. **Стандарт шифрования данных**

В 1977 году в США был предложен стандарт для шифрования данных — DES (Data Encryption Standard), разработанный в IBM. В 1980 он был одобрен ведущей мировой организацией по стандартам — ANSI. В настоящее время алгоритм DES широко используется для защиты коммерческой информации.

DES — это классическая криптосистема с открытым способом шифровки и дешифровки, секретность которой обеспечивается исключительно ключом. Основные достоинства DES:

- используется только один ключ фиксированной длины 56 бит (в системах с открытым ключом длина ключа должна быть более 300 бит);
- зашифровав сообщение с помощью одной программы, для расшифровки можно использовать другую;
- относительная простота алгоритма обеспечивает высокую скорость работы (как минимум, на порядок выше скорости работы алгоритма для криптосистемы с открытым ключом);
- достаточно высокая стойкость алгоритма (стойкость конкретного зашифрованного сообщения зависит от выбора ключа).

Главный недостаток DES связан с его классической организацией, т. е. с необходимостью обеспечивать сверхнадежный канал для передачи ключей.

Алгоритм DES предназначен для шифровки ровно 64 бит исходных данных — более длинные сообщения должны разбиваться на части длиной 64 бита, а более короткие дополняться нулями или пробелами. Собственно шифровка и дешифровка обеспечивается многократными битовыми перестановками в исходном сообщении, определяемыми стандартными перестановочными матрицами и ключом.

Примером программы, реализующей алгоритм DES, является программа DISKREET из пакета Norton Utilities.

33. **Информация в Internet**

Самый распространенный тип данных в компьютерном мире — это *текстовые* файлы, которые непосредственно в той или иной мере понятны для человека, в отличие от *бинарных* файлов, ориентированных исключительно на компьютерные методы обработки. С использованием текстовых файлов связаны две проблемы.

Первая заключается в сложности единообразного представления символов текста. Для представления английских текстов достаточно ASCII. Для работы с другими языками на основе латинского алфавита, языками на основе кириллицы и некоторыми другими нужно уже несколько десятков наборов расширенного ASCII. Это означает, что

одному и тому же коду, большему 127, в каждом наборе соответствует свой символ. Ситуацию усложняет и то, что для некоторых языков, в частности, русского существует несколько наборов ASCII+. Кроме того, необходимо, чтобы все символы каждого языка помещались в один набор, что невозможно для таких языков, как китайский или японский. Таблица кодировки Unicode, предназначенная для постепенной замены ASCII, — 16-разрядная, что позволяет представить 65536 кодов. Она широко используется в Linux и Microsoft Windows. Варианты Unicode позволяют использовать 31-разрядное кодирование. Использование Unicode требует переделки всех программ, рассчитанных для работы с текстами ASCII.

Для того, чтобы увидеть символы, соответствующие кодам из текстового файла, каждому коду нужно сопоставить визуальное представление символа из выбранного шрифта.

Компьютерный шрифт — это набор именованных кодами рисунков знаков.

Таким образом, чтобы интерактивно работать с текстовым файлом необходимо знать его кодировку (из текстовых файлов, как правило, прямой информации о кодировке получить нельзя — ее надо знать или угадать!) и иметь в системе шрифт, соответствующий этой кодировке.

Вторая проблема связана с тем, что такие средства как курсивный, полужирный или подчеркнутый текст, а также графики, диаграммы, примечания, звук, видео и т. п. элементы электронных документов, выходят за рамки естественных, интуитивных элементов текста и требуют соглашений по их использованию, что приводит к возникновению различных форматов текстовых данных. Последние иногда даже не ориентированы на непосредственную работу с ними человека, фактически не отличаясь по назначению в таких случаях, от бинарных данных.

Внесение в простой текст (plain text) дополнительной информации об его оформлении или структуре осуществляется при помощи *разметки текста* (markup). Различают *физическую* или *процедурную* разметку и *логическую* или обобщенную разметку.

При физической разметке точно указывается, что нужно сделать с выбранным фрагментом текста: показать курсивным, приподнять, центрировать, сжать, подчеркнуть и т. п. При логической разметке указывается структурный смысл выбранного фрагмента: примечание, начало раздела, конец подраздела, ссылка на другой фрагмент и т. п.

Для печати документа на принтере или показе на экране используется физическая разметка. Исторически она появилась первой, но имеет очевидные недостатки. Например, в Америке и Европе существуют разные стандарты на размер писчей бумаги, наборы шрифтов и размер экрана меняются от системы к системе, — подобные обстоятель-

ства требуют трудоемкого изменения физической разметки текста при использовании одного и того же документа на разных компьютерах. Кроме того, физическая разметка, как правило, привязана к конкретным программным средствам, время жизни которых ограничено, что не позволяет вести архивы документации без риска через несколько десятков лет остаться без средств для работы с ними.

Логическую разметку всегда можно преобразовать в физическую, используя *таблицу стилей*, которая представляет собой перечисление способов отображения каждого логического элемента. Таким образом, имея наборы документов в логической разметке можно всегда при печати придавать им наиболее привлекательный вид, своевременно получая от специалистов-дизайнеров новейшие таблицы стилей. Преобразование физической разметки в логическую формальными средствами практически невозможно.

Основные форматы текста с разметкой:

1) HTML — Hyper Text Markup Language, язык разметки гипертекста;
2) XML — eXtensible Markup Language, расширяемый язык разметки;
3) SGML — Standard Generalized Markup Language, стандартный язык обобщенной разметки;
4) TEX;
5) PostScript;
6) PDF — Portable Document Format, формат для переносимых документов, или Acrobat (частично бинарный).

Документы в Internet часто публикуются в обработанном программами сжатия данных виде. Наиболее используемые форматы сжатия — это zip и tgz (tar.gz). Формат tgz — это результат конвейерного применения команд сначала tar (собирает файлы и каталоги в один файл с сохранением структуры каталогов) и затем gzip.

Часто в Internet нужно преобразовывать бинарные данные в текстовые (для отправке по электронной почте, например) и затем наоборот. Для этого, в частности, служат программы uuencode (перевести в текст) и uudecode (перевести из текста). В текстовом файле закодированный текстом бинарный файл помещается между словами, начинающими строки, begin и end. Строка begin должна содержать атрибуты и имя бинарного файла.

34. **HTML, XML и SGML**

World Wide Web (WWW, всемирная паутина) базируется на трех стандартах: URI (Universal Resource Identifier, универсальный идентификатор ресурса, раньше назывался URL) — предоставляет стандартный способ задания местоположения любого ресурса Internet, HTTP

(Hyper Text Transfer Protocol, протокол передачи гипертекста), HTML — язык страниц WWW.

HTML — язык логической разметки, хотя и допускающий возможность *рекомендовать* ту или иную физическую разметку выбранного фрагмента текста. Конкретная физическая разметка документа зависит от программы-браузера (browser), используемой для его просмотра. Документы HTML из-за содержащихся в них, как правило, большого количества ссылок на другие документы HTML, с которыми они образуют единое целое, мало приспособлены для распечатки на принтере.

Имя файла с документом HTML имеет обычно расширение html или htm. Существуют ряд программ, позволяющих создавать документы HTML в визуальном режиме и не требующих от их пользователя знания HTML. Но создать сложный интерактивный документ без такого знания непросто.

Элементы разметки HTML состоят из *тегов* (tag). Теги заключаются в угловые скобки, у них, как правило, есть имя и они могут иметь дополнительные атрибуты. Например, тег ⟨A HREF="http://www.linux.org"⟩ имеет имя A (anchor, якорь), атрибут HREF со значением "http://www.linux.org".

Некоторые теги самодостаточны, например, тег разрыва строки ⟨BR⟩ (break), но большинство тегов — это пары из *открывающего* (start tag) и *закрывающего* (end tag) тегов. Имя закрывающего тега отличается от имени открывающего только тем, что перед ним ставится наклонная черта (slash). Например, если имя открывающего тега A, то имя закрывающего — /A. Открывающий и закрывающий теги обрамляют некоторый фрагмент текста, вместе с которым они образуют *элемент текста*. Элементы текста могут быть вложенными.

Парные теги EM (emphasis, выделение), STRONG (особо выделить), CITE (цитата или ссылка), CODE (компьютерная программа), SAMP (sample, текст примера), STRIKE (зачеркнуть) и некоторые другие позволяют логически выделить фрагменты текста, а парные теги B (bold, полужирный), I (italic, курсив), U (undelined, подчеркнутый), TT (typewriter, пишущая машинка), SUB (subscript, нижний индекс), SUP (superscript, верхний индекс) и другие — рекомендовать физически выделить фрагмент текста указанным образом.

Полный документ представляет собой один элемент текста HTML. Заголовки — это элементы H1, H2, H3 и т. д. Число после H (header) — это уровень вложенности заголовка, т. е. H1 — это заголовок всего документа, H2 — заголовок раздела документа, H3 — подраздела и т. д. Абзацы — это элементы P (paragraph). Элементы PRE (preformatted) должны отображаться браузером с таким же разбиением на строки как и в исходном документе.

Специальные символы можно ввести в документ, используя их име-

на (entity), заключенные между знаками & и точка с запятой. Например, сам знак & можно ввести как &, а знак кавычка — ".

Ссылки и маркеры, объявляются при помощи атрибутов HREF и NAME соответственно. Например, элемент ⟨A NAME="chapter3"⟩⟨/A⟩ — это метка, на которую можно ссылаться по имени chapter3, используя, например, ссылку ⟨A HREF="#chapter3"⟩Глава 3⟨/A⟩.

Тег IMG (image, образ) позволяет вставить графическую картинку в документ, используя два основных атрибута: SRC (source, источник) для указания URI файла с графикой и ALT (alternative, альтернатива) для указания альтернативного текста, показываемого вместо картинки, в случае, когда файл с графикой недоступен или его тип неизвестен браузеру.

Документы HTML могут быть использованы для интерактивной работы. Например, элемент FORM позволяет пользователю web-страницы передать введенную в страницу информацию на HTTP-сервер. Элемент FORM может содержать разнообразные кнопки, списки, всплывающие меню, однострочные и многострочные текстовые поля и другие компоненты. Обработкой введенных переданных на сервер данных и созданием динамических HTML-документов в ответ на них занимаются специальные программы, CGI-скрипты (common gate interface), установленные на сервере.

Комментарии вводятся между символами ⟨!-- и --⟩.

HTML содержит средства для описания данных в виде таблиц и использования таблиц стилей. HTML использует стандартные системные шрифты, т.е. не существует шрифтов специально для www-страниц.

Имена файлов-документов SGML, как правило, имеют расширение sgml. SGML с начала 1970-х разрабатывался фирмой IBM, а с 1986 года принят в качестве международного стандарта (ISO 8879) для формата документов с логической разметкой. Сначала документ SGML содержит описание вида кодирования и разметки текста и затем сам размеченный текст. HTML — это SGML с фиксированной разметкой. Создатели технологии WWW отказались от полной поддержки SGML только потому, что в начале 1990-х системы, которые могли работать с SGML в реальном времени были очень дороги.

Элементы SGML делятся на четыре категории:

1) описательные маркеры — определяют структуру документа — им соответствуют элементы разметки HTML типа H1, P, A, IMG и т.п.;
2) ссылки на данные — им соответствуют элементы разметки HTML типа &
3) описательные конструкции компонент документа в их структурной взаимосвязи — они не входят в HTML, но определяют его. Их рекомендуется начинать с комбинации знаков ⟨! и заканчивать знаком

⟩. Примером конструкции, определяющей ссылку &ref; на словосочетание “The Reference” будет ⟨!ENTITY ref "The Reference"⟩;

4) инструкции по обработки текста — их рекомендуется заключать между знаками ⟨? и ⟩ — они вводят элементы текста, ориентированного на конкретную, зависящую от системы обработку. В HTML с их помощью, например, вставляют код для обработки на сервере WWW страниц.

Документы SGML можно конвертировать как в гипертекст, так и в любой формат, ориентированный на распечатку, например, TEX или Microsoft Word. Ведение документации в формате SGML во многих отношениях оптимально.

С 1996 официально идет разработка формата XML — подмножества SGML, которое предполагается использовать в Internet наряду с HTML. Преимущество XML перед HTML в его четкой связи с SGML, что позволяет стандартным образом вводить в документ новые конструкции, избегая тем самым неконтролируемого введения в язык новых возможностей, как это происходит с HTML.

▶ Упражнение 51
Как на HTML описать заголовок первого уровня “Глава 2”, на который можно будет ссылаться по имени “2”?

35. TEX

Известный американский математик и теоретик программирования Дональд Кнут (D. E. Knuth) более 10 лет с конца 1970-х годов разрабатывал систему верстки книг TEX (произносится “тех”). Существует множество расширений возможностей базового (plain) TEX. TEX популярен прежде всего в академических кругах, т. к. в целом он весьма сложен для изучения. В отличие от систем, ориентированных на интерпретацию разметки, подобных Microsoft Word или Sun Star Writer, TEX — компилирующая система. Результат компиляции документа TEX — это файл в бинарном формате dvi (device independent), который можно, используя драйверы конкретных устройств (принтеров, экрана), распечатать. TEX использует собственную систему масштабируемых шрифтов, которые масштабируются не в реальном времени интерпретацией как шрифты True Type или PostScript, а компиляцией при помощи программы METAFONT. В Internet доступны тексты программ TEX и METAFONT — они написаны на Паскале. Шрифты METAFONT написаны на специальном языке, с декларативным синтаксисом. TEX позволяет также использовать шрифты True Type и Adobe Type 1 и Type 3. Прочитать и понять содержимое документа TEX несложно, но скомпилировать и распечатать, а тем более создать новый документ без помощи специалиста или основательной подготовки не-

просто. Однако TEX до сих пор является почти единственной доступной бесплатно системой, позволяющей получать документы типографского качества. В plain TEX используется физическая разметка, а в наиболее популярном его расширении LATEX также и логическая. TEX — это язык макросов, большинство из которых начинаются с символа обратная косая черта и состоят затем из букв. Например, запись в документе plain TEX \centerline{Это {\it мой} заголовок} означает центрировать строку-абзац "Это мой заголовок", напечатав слово "мой" в нем курсивом, а запись $$\int_1^x{dt\over t}=\ln x$$ — формулу

$$\int_1^x \frac{dt}{t} = \ln x.$$

TEX — это особый язык программирования. Энтузиасты TEX написали на нем интерпретатор языка Бэйсик. Документы TEX могут иметь очень сложную структуру и из-за этого их в общем случае нельзя конвертировать в другие форматы. Документы HTML или Microsoft Word теоретически можно всегда конвертировать в формат TEX.

Система GNU texinfo основана на TEX, но использует совершенно другой набор макросов. Макросы в этой системе должны начинаться со знака @. Документы texinfo можно преобразовать как в документ HTML, так и в качественную распечатку. В отличие от SGML, средства для такого преобразования — это часть системы texinfo. Возможности texinfo для верстки документов несколько ограниченней по сравнению с другими развитыми TEX-системами.

Расширения имен файлов документов TEX — tex; LATEX — tex, latex, ltx, sty (стили) и др.; METAFONT — mf (исходные программы шрифтов), tfm (метрики шрифтов, нужны на этапе компиляции документа TEX), pk (матрицы шрифтов, нужны при печати dvi-файла); texinfo — texi, texinfo.

36. **PostScript и PDF**

PostScript — это универсальный язык программирования (имеет много общего с языками Форт и Лисп), предоставляющий большой набор команд для работы с графикой и шрифтами. Он является фактическим международным стандартом издательских систем. Разрабатывается фирмой Adobe Systems с первой половины 1980-х. Используется, как встроенный язык принтеров для высококачественной печати, а также некоторыми системами X Window при выводе данных на экран дисплея. Существуют и программы-интерпретаторы языка PostScript. Лучшая из них — это Ghostscript. Программа GhostView предоставляет удобный оконный интерфейс для Ghostscript и существует для большинства ОС.

PostScript-программы можно писать вручную, но обычно текст PostScript генерируется автоматически программами вывода данных. Расширения имен файлов с PostScript-программой — это, как правило, ps, eps (Encapsulated PostScript, файл-картинка с заданными размерами), pfa (шрифт), pfb (бинарное представление pfa), afm (метрики шрифта, могут быть частично получены из соответствующего pfa-файла), pfm (бинарное представление afm).

Преимущество формата PostScript в том, что он, как и формат DVI, независим от физических устройств воспроизведения. Один и тот же PostScript-файл можно выводить как на экран с разрешением 72 dpi (dot per inch, точек на дюйм) или лазерный принтер разрешением 600 dpi, так и на типографскую аппаратуру с разрешением 2400 dpi, имея гарантии, что изображение будет наилучшего качества, возможного на выбранной аппаратуре. Возможности PostScript перекрывают возможности DVI, поэтому некоторые TEX-системы при компиляции документов производят сразу файлы в формате PostScript или PDF.

Файлы PostScript можно вручную корректировать, но из-за сложности языка — это очень не просто, особенно если используются символы, не входящие в ASCII. Фактически эти файлы можно рассматривать как “только для чтения” и использовать для распространения информации, не подлежащей изменению. Комментарии в PostScript, как и в TEX, начинаются знаком % и заканчиваются концом строки. Первая строчка PostScript-программы обычно содержит точное название формата файла. Собственно программа начинается в файле с символов %! и заканчивается символами %%EOF. PostScript-программы кроме собственной системы шрифтов могут использовать шрифты True Type фирм Apple и Microsoft.

Различают уровни (levels) языка PostScript. Уровень 1 может поддерживать только черно-белую графику. Уровень 2 может работать с цветом. Уровень 3 — это современное состояние языка.

Данные из файла PostScript можно показывать по мере их поступления, что удобно для использования в Internet. Однако есть две причины, по которым документы PostScript сравнительно редко включаются в web-страницы:

1) они весьма велики по размерам (этот недостаток снимается программами сжатия, работающими в реальном времени);
2) они могут содержать в себе шрифты, защищенные авторскими правами (шрифты их владелец может использовать при печати, но не распространять).

Файлы в формате PDF лишены двух означенных недостатков: они сжаты и из них сложно извлечь отдельные шрифты, — поэтому они стали фактическим стандартом Internet для обмена документами, не подлежащими изменению. Программы для просмотра PDF-файлов до-

ступны бесплатно. Наиболее используемая из них — это Adobe Acrobat Reader. Первая строчка файла в формате PDF начинается со знака %, за которым следует идентификационная запись версии формата PDF, используемой в этом файле. Далее, как правило, идут бинарные данные. Расширение имени PDF-файла — pdf.

Между документами PostScript и PDF можно осуществлять взаимно-однозначное преобразование, хотя PDF в отличие от PostScript — это не язык программирования, а скорее язык описания документа.

Приложение А. **Ответы на все упражнения**

1. 87 и 119.

2. 24 КГц.

3. 8192.

4. $x = 5$.

5. $HX = 0.9 + \log_2 5 - 0.3\log_2 3 \approx 2.75$ бит/сим.

6. $I(Y, X_1) = 0.5$ бит/сим.

7. $I(Z, X_1) = I(X_1, X_1) = HX_1 = 1$ бит/сим, т. е. Z полностью определяет X_1 и, следовательно, X_1 — это функцией от Z. $HZ = 2$ бит/сим.

8. $I(X_1, X_2) = (5 - 3\log_2 3)/3 \approx 0.08$ бит/сим.

9. $I(X_1, Y) = (10 - 3\log_2 3)/8 \approx 0.66$ бит/сим, $HX_1 = 2$ бит/сим, $HY = (26 - 3\log_2 3)/8 \approx 2.65$ бит/сим.

10. $I(Z, X_1) = (22 - 3\log_2 3)/16 \approx 1.08$ бит/сим, $HZ = (54 - 3\log_2 3)/16 \approx 3.08$ бит/сим.

11. $I(X_1, Y) = (3\log_2 3 - 2)/9 \approx 0.31$ бит/сим, $I(X_2, Y) = (3\log_2 3 + 4)/9 \approx 0.97$ бит/сим, $HX_1 = HX_2 = \log_2 3 \approx 1.58$ бит/сим, $HY = (12\log_2 3 - 2)/9 \approx 1.89$ бит/сим.

12. $HX = 7/4 = 1.75$ бит/сим, $HY = (24 - 3\log_2 3 - 5\log_2 5)/8 \approx 0.95$ бит/сим, $HZ = (328 - 12\log_2 3 - 35\log_2 5 - 17\log_2 17)/64 \approx 2.47$ бит/сим, $I(X, Y) = (216 - 12\log_2 3 - 35\log_2 5 - 17\log_2 17)/64 \approx 0.72$ бит/сим.

13. $ML1(X) = 3$ бит/сим, $ML2, 3, 4(X) = 2.2$ бит/сим, $HX = \log_2 5 - 0.2 \approx 2.12$ бит/сим.

14. $code(0) = 10$, $code(1) = 0$, $code(2) = 11$ — это один из вариантов кодирующей функции. $ML(X) = HX = 1.5$ бит/сим.

15. $code(2^n) = \underbrace{1\cdots1}_{n-1}0$ или $code(2^n) = \underbrace{0\cdots0}_{n-1}1$. $HX = \sum\limits_{n=1}^{\infty} n/2^n = ML(X) = 2$ бит/сим.

16. $ML(X) \geqslant HX \approx 3.25$ бит/сим.

17. $inf(s1) = 1$, $cont(s1) = 2$, $inf(s2) = 0.5$, $cont(s2) = 0.75$.

18. 1.56 бит/сим.

19. $HX \approx 2.17$ бит/сим, код Хаффмена $ML(X) \approx 2.22$ бит/сим, код Шеннона-Фэно $ML(X) \approx 2.28$ бит/сим.

20. Шеннона-Фэно, Хаффмена: $ML_1(X_1) = 2$ бит/сим., $ML_1(X_2) = 2.25$ бит/сим., $ML_1(X_3) = 2.7$ бит/сим., $ML_1(X_4) = 2^{13}/_{60}$ бит/сим. Арифметический: $ML_1(X_1) = 1^5/_6$ бит/сим., $ML_1(X_2) = 2.05$ бит/сим., $ML_1(X_3) = 2.3$ бит/сим., $ML_1(X_4) = 2^1/_{60}$ бит/сим.

21. $L_{\text{Хаффмена}} = 3$ бита, $L_{\text{арифметический}} = 4$ бита.

22. 010001011, 01011111.

23. 81, в 27 раз.

24. Считая, что код генерирутся д. с. в. X с распределением $P(X = \text{A}) = 2/3$, $P(X = \text{B}) = 1/3$, можно получить наилучшие коды, для которых $L_{\text{Хаффмена-1}}(\text{ABAAAB}) = 6$ бит, $L_{\text{Хаффмена-2}}(\text{ABAAAB}) = 5$ бит, $L_{\text{Хаффмена-3}}(\text{ABAAAB}) = 5$ бит, $L_{\text{арифметический}}(\text{ABAAAB}) = 1$ бит

25. 'B'10'C'1101

26. *code*(AABCDAACCCCCDBB) = 'A'10'B'00'C'000'D'00011001111 100110011001, L(AABCDAACCCCCDBB) = 62 бит, длина исходного сообщения — 112 бит. *code*(КИБЕРНЕТИКИ) = 'К'0'И'00'Б'100'Е'000'Р'1 100'Н'1111000'Т'100110111, L(КИБЕРНЕТИКИ) = 85 бит, длина исходного сообщения — 88 бит. *code*(СИНЯЯ СИНЕВА СИНИ) = 'С'0'И' 00'Н'100'Я'001100' '101001011100'Е'11000'В'10100'А'1010101101101111, L(СИНЯЯ СИНЕВА СИНИ) = 114 бит, длина исходного сообщения — 136 бит.

27. Распакованное сообщение — AFXAFFXFXAXAFFA, его длина — 120 бит, длина сжатого кода — 52 бит.

28. 01000010111001.

29. AABCDAACCCCCDBB, LZ77: ⟨0,0,'A'⟩ ⟨11,1,'B'⟩ ⟨0,0,'C'⟩ ⟨0,0, 'D'⟩ ⟨7,2,'C'⟩ ⟨11,2,'C'⟩ ⟨5,2,'B'⟩ ⟨0,0,'B'⟩, длина $8 * 15 = 120$ бит; LZSS: 0'A'1⟨11,1⟩0'B'0'C'0'D'1⟨7,2⟩1⟨8,1⟩1⟨11,1⟩1⟨10,2⟩1⟨5,1⟩ 1⟨3,1⟩1⟨11,1⟩, длина $8 * 7 + 4 * 9 = 92$ бит; LZ78: ⟨0,'A'⟩ ⟨1,'B'⟩ ⟨0,'C'⟩ ⟨0,'D'⟩ ⟨1,'A'⟩ ⟨3,'C'⟩ ⟨6,'D'⟩ ⟨0,'B'⟩ ⟨0,'B'⟩, длина $9 * 12 = 108$ бит; LZW: 0'A'0'A'0'B'0'C'0'D' ⟨256⟩0'C'⟨262⟩⟨259⟩0'B'0'B', длина $11 * 9 = 99$ бит. КИБЕРНЕТИКИ, LZ77: ⟨0,0,'К'⟩ ⟨0,0,'И'⟩, ⟨0,0,'Б'⟩ ⟨0,0,'Е'⟩ ⟨0,0,'Р'⟩ ⟨0,0,'Н'⟩ ⟨9,1,'Т'⟩ ⟨5,1,'К'⟩ ⟨0,0,'И'⟩, длина $9 * 15 = 135$ бит; LZSS: 0'К'0'И'0'Б'0'Е'0'Р'0'Н' 1⟨9,1⟩0'Т'1⟨5,1⟩1⟨5,2⟩, длина $3 * 7 + 7 * 9 = 84$ бит; LZ78: ⟨0,'К'⟩ ⟨0,'И'⟩ ⟨0,'Б'⟩ ⟨0,'Е'⟩ ⟨0,'Р'⟩ ⟨0,'Н'⟩ ⟨4,'Т'⟩ ⟨2,'К'⟩ ⟨0,'И'⟩, длина $9 * 12 = 108$ бит; LZW: 0'К'0'И'0'Б'0'Е'0'Р'0'Н'0'Е'0'Т'0'И'⟨256⟩, длина $10 * 9 = 90$ бит. "СИНЯЯ СИНЕВА СИНИ", LZ77: ⟨0,0,'С'⟩ ⟨0,0,'И'⟩ ⟨0,0,'Н'⟩ ⟨0,0,'Я'⟩ ⟨11,1' '⟩ ⟨6,3,'Е'⟩ ⟨0,0,'В'⟩ ⟨0,0,'А'⟩ ⟨5,4,'И'⟩, длина $9 * 15 = 135$ бит; LZSS: 0'С'0'И'0'Н'0'Я'1⟨11,1⟩0' '⟨6,3⟩0'Е'0'В'0'А'1⟨5,4⟩1⟨10,1⟩, длина $4 * 7 + 8 * 9 = 100$ бит; LZ78: ⟨0,'С'⟩ ⟨0,'И'⟩ ⟨0,'Н'⟩ ⟨0,'Я'⟩ ⟨4,' '⟩ ⟨1,'И'⟩ ⟨3,'Е'⟩ ⟨0,'В'⟩ ⟨0,'А'⟩ ⟨0,' '⟩ ⟨6,'Н'⟩ ⟨0,'И'⟩, длина $12 * 12 = 144$ бит; LZW: 0'С'0'И'0'Н'0'Я'0'Я'0' '⟨256⟩0'Н'0'Е'0'В'0'А'⟨261⟩⟨257⟩0'И', длина $14 * 9 = 126$ бит.

30. Нет. Это следует из очевидного неравенства для длин кодов $\log_2(L_D + 256) < \log_2(L_D) + 8$, где L_D — это размер словаря.

31. Во всех случаях сообщение — AFXAFFXFXAXAFFA, длина кода LZ77 — 105 бит, LZSS — 62 бит, LZ78 — 108 бит, LZW — 99 бит.

32. 2000 бод.

33. 1) $8000/3 \approx 2666.67$ сим/сек; 2) ≈ 2523 сим/сек; 3) 2000 сим/сек.

34. Пусть X — д. с. в., определяющая передатчик, а Y — д. с. в., определяющая приемник. Тогда $P(Y = 00/X = 00) = pp$, $P(Y = 00/X = 01) = pq$, ..., $P(Y = 00/X = 11) = qq$, ...

35. $C_{14}^5 p^9 q^5$, $\sum_{i=0}^{4} C_{14}^i p^{14-i} q^i$, $\sum_{i=0}^{4} C_{14}^i = 1471$.

36. $\approx 0.3\%$, $\approx 7.7\%$; $\approx 0.004\%$, $\approx 0.797\%$.

37. $r = 6$, $11 \leqslant r \leqslant 16$.

38. $r \geqslant 2$, $r \geqslant 9$.

39. E_1: 1. 00 → 00000, 01 → 01110, 10 → 10101, 11 → 11011; 2. $\min d = 3$, $P_{\text{необнаружения ошибки}} = 2p^2q^3 + pq^4$, код исправляет или обнаруживает все ошибки кратности соответственно до 1 или 2;

3. 00000 01110 10101 11011
   00001 01111 10100 11010
   00010 01100 10111 11001
   00100 01010 10001 11111
   01000 00110 11101 10011
   10000 11110 00101 01011
   00011 01101 10110 11000
   10010 11100 00111 01001;

4. $P_{\text{правильной передачи}} = p^5 + 5p^4q + 2p^3q^2$, код исправляет все ошибки кратности 1 и 2 из 10 ошибок кратности 2; 5. 10001 → 10, 01110 → 01, 10101 → 10. E_2: 1. 000 → 0000, 001 → 0010, 010 → 0101, 011 → 0111, 100 → 1001, 101 → 1011, 110 → 1100, 111 → 1110; 2. $\min d = 1$, $P_{\text{необнаружения ошибки}} = p^3q + 3p^2q^2 + 3pq^3$, код не исправляет и не обнаруживает все ошибки никакой кратности;

3. 0000 0010 0101 0111 1001 1011 1100 1110
   0001 0011 0100 0110 1000 1010 1101 1111;

4. $P_{\text{правильной передачи}} = p^4 + p^3q$, код исправляет 1 из 4 ошибок кратности 1; 5. 1001 → 100, 0110 → 011, 1101 → 110.

40. нет, т.к. $\sum_{i=0}^{2} C_{14}^i \neq 2^8$.

41. $55_{10} = 001010101 \to 0001001010111$, $200_{10} \to 100011001000$, $1000001000001 \to 000100101$, $1100010111100 \to 001011101$.

42. 0100 → 01100010100, 10001101 → 110011101011001, 11110 → 10011110110.

43. Первое — нет, второе — да.

44. $g(x) = 1 + x + x^2 + x^4 + x^5 + x^8 + x^{10}$.

45. 1000, 1111.

46. ПТУРХЧЧЮНФЫ.

47. 22: A отправляет B 58, B возвращает 94, A окончательно отправляет 115; 17: B отправляет A 135, A возвращает 15, B окончательно отправляет 143.

48. 53, 51; 247.

49. для $a = 33$ $\alpha = 237$.

50. $\alpha = 21$, 124.

51. ⟨H1⟩⟨A name="2"⟩⟨/A⟩Глава 2⟨/H1⟩.

Приложение Б. **Управляющие коды ASCII**

Код			Полное имя кода в Unicode (краткое имя в ASCII)
10-й	16-й	Клавиатурный	
Перевод имени кода — описание использования кода.			

Выше представлен шаблон для следующей далее таблицы управляющих символов. Под клавиатурным кодом подразумевается комбинация двух клавиш, Ctrl (Control, в таблице это знак ^) и приводимой, одновременное нажатие которых должно производить соответствующий код.

0	00	^@	NULL (NUL)
Пусто — этот код используется как завершающий в представлении строк многими системами программирования, например, Си, поэтому его использование в текстовых файлах крайне нежелательно.			
1	01	^A	START OF HEADING (SOH)
Начало заголовка — практически не используется.			
2	02	^B	START OF TEXT (STX)
Начало текста — практически не используется.			
3	03	^C	END OF TEXT (ETX)
Конец текста — в Unix и MS-DOS ввод этого символа с клавиатуры служит сигналом для прекращения выполнения программы.			
4	04	^D	END OF TRANSMISSION (EOT)
Конец передачи — в Unix и PostScript означает конец вводимых данных.			
5	05	^E	ENQUIRY (ENQ)
Кто там? — практически не используется.			
6	06	^F	ACKNOWLEDGE (ACK)
Подтверждение, да — практически не используется.			
7	07	^G	BELL (BEL)
Звонок — при его печати на консоли MS-DOS или Unix должен производиться звуковой сигнал.			
8	08	^H	BACKSPACE (BS)
Возврат на шаг — означает, что следующий символ следует печатать с предшествующей позиции.			
9	09	^I	HORISONTAL TABULATION (TAB)
Горизонтальная табуляция — переход на следующую позицию табуляции.			
10	0A	^J	LINE FEED (LF)
Подача новой строки — переход на новую строку. В текстовых файлах MS-DOS и Microsoft Windows с сохранением текущей горизонтальной позицию. В текстовых файлах Unix с переходом на первую горизонтальную позицию.			

11	0B	^K	VERTICAL TABULATION (VT)
Вертикальная табуляция — используется очень редко, как правило, принтерами.			
12	0C	^L	FORM FEED (FF)
Подача новой формы — для консоли, как правило, означает очистку экрана, для принтера — завершение печати на текущем листе и запрос нового.			
13	0D	^M	CARRIAGE RETURN (CR)
Возврат каретки — переход на первую горизонтальную позицию строки. В текстовых файлах MS-DOS и Microsoft Windows с сохранением текущей строки, а в текстовых файлах Macintosh OS с переходом на новую строку. В текстовых файлах Unix не используется.			
14	0E	^N	SHIFT OUT (SO)
Выход — используется очень редко, как правило, принтерами.			
15	0F	^O	SHIFT IN (SI)
Вход — используется очень редко, как правило, принтерами.			
16	10	^P	DATA LINK ESCAPE (DLE)
Авторегистр 1 — практически не используется.			
17	11	^Q	DEVICE CONTROL ONE (DC1)
Используется некоторыми телекоммуникационными протоколами как байт X-ON.			
18	12	^R	DEVICE CONTROL TWO (DC2)
Практически не используется.			
19	13	^S	DEVICE CONTROL THREE (DC3)
Используется некоторыми телекоммуникационными протоколами как байт X-OFF.			
20	14	^T	DEVICE CONTROL FOUR (DC4)
Практически не используется.			
21	15	^U	NEGATIVE ACKNOWLEDGE (NAK)
Нет — практически не используется.			
22	16	^V	SYNCHRONOUS IDLE (SYN)
Синхронизация — практически не используется.			
23	17	^W	END OF TRANSMISSION BLOCK (ETB)
Конец блока — практически не используется.			
24	18	^X	CANCEL (CAN)
Аннулирование — используется очень редко, как правило, принтерами.			
25	19	^Y	END OF MEDIUM (EM)
Конец носителя — практически не используется.			
26	1A	^Z	SUBSTITUTE (SUB)
Замена — в MS-DOS, Macintosh OS и CP/M — это маркер конца текстового файла.			

27	1B	^[	ESCAPE (ESC)
Авторегистр 2 — указывает на то, что некоторое количество кодов после него и он сам образуют группу, рассматриваемую как один код.			
28	1C	^\	FILE SEPARATOR (FS)
Разделитель файлов — практически не используется.			
29	1D	^]	GROUP SEPARATOR (GS)
Разделитель групп — практически не используется.			
30	1E	^^	RECORD SEPARATOR (RS)
Разделитель записей — практически не используется.			
31	1F	^_	UNIT SEPARATOR (US)
Разделитель элементов — практически не используется.			
127	7F		DELETE (DEL)
Забой — удаление последнего видимого знака печатаемой строки.			

В “чисто” текстовых (plain text) файлах допустимы только управляющие символы, отмечающие концы строк и, как правило, переходы на позиции табуляции (код 9). Маркер конца строки в Unix — это код 10, в Macintosh OS — 13, в CP/M, MS-DOS и Microsoft Windows — последовательность 13, 10.

Приложение В. **Кодировка видимых символов ASCII**

Код		Символ	Имя символа в Unicode 3.2
10-й	16-й		
32	20		SPACE
33	21	!	EXCLAMATION MARK
34	22	"	QUOTATION MARK
35	23	#	NUMBER SIGN
36	24	$	DOLLAR SIGN
37	25	%	PERCENT SIGN
38	26	&	AMPERSAND
39	27	´	APOSTROPHE
40	28	(	LEFT PARENTHESIS
41	29	)	RIGHT PARENTHESIS
42	2A	*	ASTERISK
43	2B	+	PLUS SIGN
44	2C	,	COMMA
45	2D	-	HYPHEN-MINUS
46	2E	.	FULL STOP
47	2F	/	SOLIDUS
48	30	0	DIGIT ZERO
49	31	1	DIGIT ONE
50	32	2	DIGIT TWO
51	33	3	DIGIT THREE
52	34	4	DIGIT FOUR
53	35	5	DIGIT FIVE
54	36	6	DIGIT SIX
55	37	7	DIGIT SEVEN
56	38	8	DIGIT EIGHT
57	39	9	DIGIT NINE
58	3A	:	COLON
59	3B	;	SEMICOLON
60	3C	<	LESS-THAN SIGN
61	3D	=	EQUALS SIGN
62	3E	>	GREATER-THAN SIGN
63	3F	?	QUESTION MARK

Код		Символ	Имя символа в Unicode 3.2
10-й	16-й		
64	40	@	COMMERCIAL AT
65	41	A	LATIN CAPITAL LETTER A
66	42	B	LATIN CAPITAL LETTER B
67	43	C	LATIN CAPITAL LETTER C
68	44	D	LATIN CAPITAL LETTER D
69	45	E	LATIN CAPITAL LETTER E
70	46	F	LATIN CAPITAL LETTER F
71	47	G	LATIN CAPITAL LETTER G
72	48	H	LATIN CAPITAL LETTER H
73	49	I	LATIN CAPITAL LETTER I
74	4A	J	LATIN CAPITAL LETTER J
75	4B	K	LATIN CAPITAL LETTER K
76	4C	L	LATIN CAPITAL LETTER L
77	4D	M	LATIN CAPITAL LETTER M
78	4E	N	LATIN CAPITAL LETTER N
79	4F	O	LATIN CAPITAL LETTER O
80	50	P	LATIN CAPITAL LETTER P
81	51	Q	LATIN CAPITAL LETTER Q
82	52	R	LATIN CAPITAL LETTER R
83	53	S	LATIN CAPITAL LETTER S
84	54	T	LATIN CAPITAL LETTER T
85	55	U	LATIN CAPITAL LETTER U
86	56	V	LATIN CAPITAL LETTER V
87	57	W	LATIN CAPITAL LETTER W
88	58	X	LATIN CAPITAL LETTER X
89	59	Y	LATIN CAPITAL LETTER Y
90	5A	Z	LATIN CAPITAL LETTER Z
91	5B	[	LEFT SQUARE BRACKET
92	5C	\	REVERSE SOLIDUS
93	5D	]	RIGHT SQUARE BRACKET
94	5E	^	CIRCUMFLEX ACCENT
95	5F	_	LOW LINE

Код		Символ	Имя символа в Unicode 3.2
10-й	16-й		
96	60	‘	GRAVE ACCENT
97	61	a	LATIN SMALL LETTER A
98	62	b	LATIN SMALL LETTER B
99	63	c	LATIN SMALL LETTER C
100	64	d	LATIN SMALL LETTER D
101	65	e	LATIN SMALL LETTER E
102	66	f	LATIN SMALL LETTER F
103	67	g	LATIN SMALL LETTER G
104	68	h	LATIN SMALL LETTER H
105	69	i	LATIN SMALL LETTER I
106	6A	j	LATIN SMALL LETTER J
107	6B	k	LATIN SMALL LETTER K
108	6C	l	LATIN SMALL LETTER L
109	6D	m	LATIN SMALL LETTER M
110	6E	n	LATIN SMALL LETTER N
111	6F	o	LATIN SMALL LETTER O
112	70	p	LATIN SMALL LETTER P
113	71	q	LATIN SMALL LETTER Q
114	72	r	LATIN SMALL LETTER R
115	73	s	LATIN SMALL LETTER S
116	74	t	LATIN SMALL LETTER T
117	75	u	LATIN SMALL LETTER U
118	76	v	LATIN SMALL LETTER V
119	77	w	LATIN SMALL LETTER W
120	78	x	LATIN SMALL LETTER X
121	79	y	LATIN SMALL LETTER Y
122	7A	z	LATIN SMALL LETTER Z
123	7B	{	LEFT CURLY BRACKET
124	7C	|	VERTICAL LINE
125	7D	}	RIGHT CURLY BRACKET
126	7E	~	TILDE

Приложение Г. **Кодировка букв русского алфавита**

В настоящее время наиболее широко используются пять (!) различных таблиц кодировки для формального представления русских букв:

I. ISO 8859-5 — международный стандарт;
II. Кодовая страница 866 (Microsoft CP866) — используется в MS-DOS;
III. Кодовая страница 1251 (Microsoft CP1251) для Microsoft Windows;
IV. На базе ГОСТ КОИ-8, koi8-r — применяется в мире Unix;
V. Unicode — используется в Microsoft Windows, Unix и клонах Unix.

Основная кодировка ГОСТ (государственный стандарт СССР) от 1987 года создана на основе рекомендаций ISO и в дальнейшем стала основой для представления знаков русских букв в Unicode. В ней и в кодировках II, III и V все буквы кроме ё и Ё расположены в алфавитном порядке. На практике эту кодировку можно встретить только на старых IBM PC совместимых компьютерах ЕС-1840 и в некоторых принтерах. Internet браузеры обычно поддерживают ее наряду с кодировками II–IV.

Кодировка CP866, разработанная на основе альтернативной кодировки ГОСТ, создана специально для ОС MS-DOS, в которой часто используются символы псевдографики. В этой кодировке эти символы имеют те же коды, что и в стандартном IBM PC совместимом компьютере.

Альтернативная кодировка ГОСТ, которая имеет два варианта, совпадает с CP866 по позициям для букв русского алфавита и знакам псевдографики. Основная кодировка ГОСТ совпадает с ISO 8859-5 только по всем знакам русских букв, кроме загланой буквы Ё.

Использование CP1251 обусловлено почти исключительно влиянием на компьютерные технологии разработок фирмы Microsoft. В ней наиболее полно по сравнению с I, II, IV представлены такие символы как ©, ®, №, различные виды кавычек и тире и т. п.

Кодировка koi8-r основана на стандартах по обмену информацией, используемых на компьютерах под управлением ОС Unix, CP/M и некоторых других с середины 1970-х. В 1993 она стандартизирована в Internet документом RFC1489.

Кодировка Unicode опирается на каталог символов UCS (Universal Character Set) стандарта ISO 10646. UCS может содержать до 2^{31} различных знаков. Коды UCS-2 — 2-байтные, UCS-4 — 4-байтные. Используются также коды переменной длины UTF-8 (Unicode Transfer Format) — 1–6-байтные, наиболее совместимые с ASCII, и UTF-16 — 2 или 4-байтные. Unicode в прикладных программах реализуется лишь частично, и в полном объеме пока нигде не поддерживается. В Linux используется UTF-8.

Достаточно широко используется кодирование на основе ASCII:

VI. На базе КОИ-7 — можно использовать при отсутствии кириллических шрифтов, код получается вычитанием 128 от соответствующего кода в koi8-r, что, как правило, дает код латинской буквы, близкой фонетически к русской.

В кодировке VI нет видимого символа для для Ъ.

Далее следует таблица, в которой представлены все перечисленные способы кодирования букв русского алфавита. В этой таблице в колонке 1 находятся символы букв, в колонке 2 часть названия букв в Unicode 3.2 (названия строчных кириллических букв начинается словами CYRILLIC SMALL LETTER, а заглавных — CYRILLIC CAPITAL LETTER, т. о., полное название буквы Д — CYRILLIC CAPITAL LETTER DE), в колонках с I по V коды десятичные и шестнадцатеричные соответствующих таблиц кодировки, а в колонке VI — символ ASCII для КОИ-7.

Кроме перечисленных можно встретить еще используемую до введения кодировок ГОСТ болгарскую кодировку, называемую также MIC, Interprog или “старый вариант ВЦ АН СССР”. На компьютерах под управлением Macintosh OS используется также своя собственная таблица кодировки для русских букв, по своему набору знаков почти совпадающая с CP1251.

1	2	I		II		III		IV		V		VI
а	A	208	D0	160	A0	224	E0	193	C1	1072	0430	A
б	BE	209	D1	161	A1	225	E1	194	C2	1073	0431	B
в	VE	210	D2	162	A2	226	E2	215	D7	1074	0432	W
г	GHE	211	D3	163	A3	227	E3	199	C7	1075	0433	G
д	DE	212	D4	164	A4	228	E4	196	C4	1076	0434	D
е	IE	213	D5	165	A5	229	E5	197	C5	1077	0435	E
ё	IO	241	F1	241	F1	184	B8	163	A3	1105	0451	#
ж	ZHE	214	D6	166	A6	230	E6	214	D6	1078	0436	V
з	ZE	215	D7	167	A7	231	E7	218	DA	1079	0437	Z
и	I	216	D8	168	A8	232	E8	201	C9	1080	0438	I
й	SHORT I	217	D9	169	A9	233	E9	202	CA	1081	0439	J
к	KA	218	DA	170	AA	234	EA	203	CB	1082	043A	K
л	EL	219	DB	171	AB	235	EB	204	CC	1083	043B	L
м	EM	220	DC	172	AC	236	EC	205	CD	1084	043C	M
н	EN	221	DD	173	AD	237	ED	206	CE	1085	043D	N
о	O	222	DE	174	AE	238	EE	207	CF	1086	043E	O
п	PE	223	DF	175	AF	239	EF	208	D0	1087	043F	P
р	ER	224	E0	224	E0	240	F0	210	D2	1088	0440	R
с	ES	225	E1	225	E1	241	F1	211	D3	1089	0441	S
т	TE	226	E2	226	E2	242	F2	212	D4	1090	0442	T
у	U	227	E3	227	E3	243	F3	213	D5	1091	0443	U
ф	EF	228	E4	228	E4	244	F4	198	C6	1092	0444	F
х	HA	229	E5	229	E5	245	F5	200	C8	1093	0445	H
ц	TSE	230	E6	230	E6	246	F6	195	C3	1094	0446	C
ч	CHE	231	E7	231	E7	247	F7	222	DE	1095	0447	^
ш	SHA	232	E8	232	E8	248	F8	219	DB	1096	0448	[
щ	SHCHA	233	E9	233	E9	249	F9	221	DD	1097	0449	]
ъ	HARD SIGN	234	EA	234	EA	250	FA	223	DF	1098	044A	_
ы	YERU	235	EB	235	EB	251	FB	217	D9	1099	044B	Y
ь	SOFT SIGN	236	EC	236	EC	252	FC	216	D8	1100	044C	X
э	E	237	ED	237	ED	253	FD	220	DC	1101	044D	\
ю	YU	238	EE	238	EE	254	FE	192	C0	1102	044E	@
я	YA	239	EF	239	EF	255	FF	209	D1	1103	044F	Q

1	2	I		II		III		IV		V		VI
А	A	176	B0	128	80	192	C0	225	E1	1040	0410	a
Б	BE	177	B1	129	81	193	C1	226	E2	1041	0411	b
В	VE	178	B2	130	82	194	C2	247	F7	1042	0412	w
Г	GHE	179	B3	131	83	195	C3	231	E7	1043	0413	g
Д	DE	180	B4	132	84	196	C4	228	E4	1044	0414	d
Е	IE	181	B5	133	85	197	C5	229	E5	1045	0415	e
Ё	IO	161	A1	240	F0	168	A8	179	B3	1025	0401	3
Ж	ZHE	182	B6	134	86	198	C6	246	F6	1046	0416	v
З	ZE	183	B7	135	87	199	C7	250	FA	1047	0417	z
И	I	184	B8	136	88	200	C8	233	E9	1048	0418	i
Й	SHORT I	185	B9	137	89	201	C9	234	EA	1049	0419	j
К	KA	186	BA	138	8A	202	CA	235	EB	1050	041A	k
Л	EL	187	BB	139	8B	203	CB	236	EC	1051	041B	l
М	EM	188	BC	140	8C	204	CC	237	ED	1052	041C	m
Н	EN	189	BD	141	8D	205	CD	238	EE	1053	041D	n
О	O	190	BE	142	8E	206	CE	239	EF	1054	041E	o
П	PE	191	BF	143	8F	207	CF	240	F0	1055	041F	p
Р	ER	192	C0	144	90	208	D0	242	F2	1056	0420	r
С	ES	193	C1	145	91	209	D1	243	F3	1057	0421	s
Т	TE	194	C2	146	92	210	D2	244	F4	1058	0422	t
У	U	195	C3	147	93	211	D3	245	F5	1059	0423	u
Ф	EF	196	C4	148	94	212	D4	230	E6	1060	0424	f
Х	HA	197	C5	149	95	213	D5	232	E8	1061	0425	h
Ц	TSE	198	C6	150	96	214	D6	227	E3	1062	0426	c
Ч	CHE	199	C7	151	97	215	D7	254	FE	1063	0427	~
Ш	SHA	200	C8	152	98	216	D8	251	FB	1064	0428	{
Щ	SHCHA	201	C9	153	99	217	D9	253	FD	1065	0429	}
Ъ	HARD SIGN	202	CA	154	9A	218	DA	255	FF	1066	042A	
Ы	YERU	203	CB	155	9B	219	DB	249	F9	1067	042B	y
Ь	SOFT SIGN	204	CC	156	9C	220	DC	248	F8	1068	042C	x
Э	E	205	CD	157	9D	221	DD	252	FC	1069	042D	|
Ю	YU	206	CE	158	9E	222	DE	224	E0	1070	042E	‘
Я	YA	207	CF	159	9F	223	DF	241	F1	1071	042F	q

Приложение Д. **Элементы теории чисел**

Каноническим разложением числа m называется разложение его на простые сомножители в виде $m = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_k^{\alpha_k}$, где $p_1, p_2, \ldots, p_k$ — все различные простые делители числа m, а $\alpha_1, \alpha_2, \ldots, \alpha_k$ — целые положительные числа.

Функцией Эйлера называется, отображение $\varphi\colon \mathbb{N} \to \mathbb{N}$,

$$\varphi(m) = p_1^{\alpha_1 - 1}(p_1 - 1) p_2^{\alpha_2 - 1}(p_2 - 1) \cdots p_k^{\alpha_k - 1}(p_k - 1),$$

$p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_k^{\alpha_k}$ — каноническое разложение m.

Например, $\varphi(2) = 1$, $\varphi(12) = \varphi(2^2 3) = 2^1(2-1)3^0(3-1) = 2 * 2 = 4$, $\varphi(1000) = \varphi(2^3 5^3) = 2^2 5^2 4 = 4 * 25 * 4 = 400$.

Числа m и n называются *взаимно простыми*, если у них нет общих делителей больших 1, т.е. НОД$(m, n) = 1$.

Функция Эйлера от числа m равна числу чисел меньших m и взаимно простых с m [7].

Для взаимно простых m и n верно равенство $\varphi(mn) = \varphi(m)\varphi(n)$ [7].

Число примитивных многочленов степени n над полем $(\mathbb{Z}_2, +, \times)$ равно $\varphi(2^n - 1)/n$ [12].

Теорема Эйлера-Ферма [7]. Для взаимно простых m и a имеет место равенство $a^{\varphi(m)} \equiv 1 \pmod{m}$.

Для решения уравнения $ax \equiv 1 \pmod{m}$, где НОД$(a, m) = 1$, можно использовать теорему Эйлера-Ферма, т.е. $x \equiv a^{\varphi(m)-1} \pmod{m}$, но это весьма трудоемкий способ. Получим решения искомого уравнения через формулу для решения эквивалентного уравнения $ax - my = 1$.

По *алгоритму Евклида* для получения НОД двух заданных чисел нужно одно число делить на другое, затем делить делитель на получаемый остаток до тех, пока остаток не станет равным нулю. Последний больший нуля остаток будет искомым НОД.

Для чисел a и m последовательность шагов алгоритма Евклида выглядит как

$$\begin{aligned} a &= mq_0 + a_1, \\ m &= a_1 q_1 + a_2, \\ a_1 &= a_2 q_2 + a_3, \\ &\ldots \\ a_{n-2} &= a_{n-1} q_{n-1} + a_n, \\ a_{n-1} &= a_n q_n, \end{aligned}$$

где $a_1, a_2, \ldots, a_n$ — остатки. Разложение $\dfrac{a}{m}$ в цепную дробь по после-

довательности частных $q_0, \ldots, q_n$ имеет вид

$$\frac{a}{m} = q_0 + \frac{a_1}{m} = q_0 + \frac{1}{\frac{m}{a_1}} = q_0 + \frac{1}{q_1 + \frac{a_2}{a_1}} = \cdots = q_0 + \cfrac{1}{q_1 + \cfrac{1}{q_2 + \cfrac{1}{q_3 + \cdots}}}.$$

Обозначим за P_k/Q_k дробь, получаемую из приведенной цепной дроби отбрасыванием членов с индексами, большими k. Например, $P_0/Q_0 = q_0$, $P_1/Q_1 = q_0+1/q_1 = (q_0q_1+1)/q_1$ и т.д. Числитель, P_k, и знаменатель, Q_k, можно вычислять рекуррентно по следующим формулам:

$$P_{-2} = 0, P_{-1} = 1, Q_{-2} = 1, Q_{-1} = 0;$$

$$\text{при } k \geqslant 0 \quad P_k = q_kP_{k-1} + P_{k-2},\ Q_k = q_kQ_{k-1} + Q_{k-2}.$$

По определению $P_n = a$ и $Q_n = m$. Кроме того,

$$F_n = P_nQ_{n-1} - P_{n-1}Q_n = (q_nP_{n-1}+P_{n-2})Q_{n-1} - P_{n-1}(q_nQ_{n-1}+Q_{n-2}) =$$
$$= -P_{n-1}Q_{n-2} + P_{n-2}Q_{n-1} = -F_{n-1} = \cdots = F_{n-2} = \cdots = (-1)^{n+1}F_{-1} =$$
$$= (-1)^{n+1}(P_{-1}Q_{-2} - P_{-2}Q_{-1}) = (-1)^{n+1}$$

или

$$(-1)^{n+1}P_nQ_{n-1} - P_{n-1}(-1)^{n+1}Q_n = 1,$$

что означает

$$a(-1)^{n+1}Q_{n-1} - m(-1)^{n+1}P_{n-1} = 1,$$

т.е. $x = (-1)^{n-1}Q_{n-1}$ и $y = (-1)^{n-1}P_{n-1}$.

Процесс получения числителей и знаменателей удобно оформить в виде таблицы:

k	-2	-1	0	1	2	$\cdots$	$n-1$	n
q_k			q_0	q_1	q_2	$\cdots$	q_{n-1}	q_n
P_k	0	1	P_0	P_1	P_2	$\cdots$	P_{n-1}	P_n
Q_k	1	0	Q_0	Q_1	Q_2	$\cdots$	Q_{n-1}	Q_n.

Таким образом, корни уравнения $ax \equiv 1 \pmod m$ вычисляются по формуле $x = (-1)^{n-1}Q_{n-1}$.

Пример. Решить уравнение $1181x \equiv 1 \pmod{1290816}$. Сначала по алгоритму Евклида получается следующая цепочка соотношений:

$$1181 = 1290816 * 0 + 1181,$$
$$1290816 = 1181 * 1092 + 1164,$$
$$1181 = 1164 * 1 + 17,$$
$$1164 = 17 * 68 + 8,$$
$$17 = 8 * 2 + 1,$$
$$8 = 1 * 8.$$

Затем составляется таблица для вычисления Q_5:

k	-2	-1	0	1	2	3	4	5
q_k			0	1092	1	68	2	8
Q_k	1	0	1	1092	1093	75416	151925	$1290816.$

Таким образом, искомый x равен 151925.

Гипотеза. Задача разложения целого числа с заданным числом разрядов на множители является труднорешаемой*.

На сегодняшний день существуют весьма быстрые алгоритмы для проверки данного числа на простоту, но для разложения 200-значного числа на множители лучшим современным компьютерам по лучшим современным алгоритмам может потребоваться миллиарды лет.

Эта гипотеза лежит в основе методов Диффи-Хеллмана.

* Задача называется *труднорешаемой*, если время ее решения зависит от объема входных данных по экспоненциальному закону и не может быть сведено к полиномиальному

Приложение Е. **Используемые обозначения**

$P(A)$ — вероятность события A.

$P(A/B)$ — вероятность события A, если известно, что событие B произошло. Условная вероятность.

$P(A, B)$ — вероятность одновременного наступления событий A и B.

$\mathbb{N}$ — множество натуральных чисел.

$\mathbb{Z}_2$ — множество из 0 и 1 — $\{0, 1\}$.

$\mathbb{R}$ — множество вещественных чисел.

$\mathbb{R}^2$ — числовая плоскость.

$\sum_i x_i$ — сумма x_i по всем возможным значениям индекса i.

$\sum_{i,j} x_{ij}$ — сумма x_{ij} по всем возможным значениям пар индексов i и j.

C_n^k — биномиальный коэффициент в формуле бинома Ньютона

$$(p+q)^n = \sum_{k=0}^{n} C_n^k p^k q^{n-k}, \quad C_n^k = \frac{n!}{k!(n-k)!}$$

или число возможных разных выборок k элементов из множества из n элементов, число сочетаний из n по k.

$\dim(\vec{X})$ — размерность вектора $\vec{X}$, число компонент $\vec{X}$.

$\#X$ — количество элементов в множестве X, мощность X.

НОД(n, m) — наибольший общий делитель n и m.

НОК(n, m) — наименьшее общее кратное n и m.

$a \equiv b \pmod{n}$ — числа a и b сравнимы по модулю n, т. е. разность $a - b$ делится на n нацело.

$f\colon A \to B$ — функция f с областью определения A и областью, содержащей все значения f, B.

$f \circ g$ — композиция функций f и g, т.е. $(f \circ g)(x) = f(g(x))$.

$(X, +, \times)$ — поле над множеством X с аддитивной операцией $+$ и мультипликативной операцией $\times$.

Приложение Ж. **Список литературы**

1. Биркгоф Г., Барти Т. *Современная прикладная алгебра* — М.: Мир, 1976.
2. Блейхер Р. *Теория и практика кодов, контролирующих ошибки* — М.: Мир, 1986.
3. Борн Г. *Форматы данных* — Киев: Торгово-издательское бюро BHV, 1995.
4. Букчин Л. В., Безрукий Ю. Л. *Дисковая подсистема IBM-совместимых персональных компьютеров* — М.: фирма “МИКАП”, 1993.
5. Винер Н. *Кибернетика* — М.: Наука, 1983.
6. Водолазкий В. *Коммерческие системы шифрования: основные алгоритмы и их реализация* //“Монитор” 6–8/92.
7. Воробьев Н. Н. *Признаки делимости* — М.: Наука, 1988.
8. Глушков В.М. *Основы безбумажной информатики* — М.: Наука, 1987.
9. Джордж Ф. *Основы кибернетики* — М.: Радио и Связь, 1984.
10. Кенцл Т. *Форматы файлов Internet* — СПб: Питер, 1997.
11. Нельсон М. *Верификация файлов* //“Журнал д-ра Добба” 1/93.
12. Нечаев В. И. *Элементы криптографии* — М.: Высшая школа, 1999.
13. Мастрюков Д. *Алгоритмы сжатия информации* //“Монитор” 7/93–6/94.
14. Питерсон Р., Уэлдон Э. *Коды, исправляющие ошибки* — М.: Мир, 1976.
15. Плотников В. *Алгоритмическая реализация криптографического метода RSA* //“Монитор” 2/94.
16. *Перспективы развития вычислительной техники: в 11 кн.: Справочное пособие/Под ред. Ю. М. Смирнова. Кн. 9.* — М.: Высшая школа, 1989.
17. Титце У., Шенк К. *Полупроводниковая схемотехника* — М.: Мир, 1983.
18. Чисар И., Кёрнер Я. *Теория информации* — М.: Мир, 1985.
19. Шеннон К. *Работы по теории информации и кибернетики* — М., Издательство иностранной литературы, 1963.
20. Яглом А., Яглом И. *Вероятность и информация* — М.: Наука, 1973.
21. *Введение в криптографию* /Под общей редакцией В. В. Ященко. — М.: МЦНМО: ”ЧеРо”, 2000.
22. *HTML 4.01 Specification* /Edited by D. Ragget, A. L. Hors, I. Jacobs — W3C: http://www.w3c.org/TR/REC-html401-19991224, 1999.
23. *The Unicode Standard, Version 3.0* — Addison Wesley Longman Publisher, 2000, ISBN 0-201-61633-5.

Приложение 3. **Предметный указатель**

Приложение И. **Именной указатель**

ОГЛАВЛЕНИЕ

Приложения

ДЛЯ ЗАМЕТОК

УЧЕБНОЕ ПОСОБИЕ

Владимир Викторович Лидовский

ТЕОРИЯ ИНФОРМАЦИИ

Редактор ?. ?. ???????

Подписано в печать ??.??.03. Заказ ???.
Формат 60×90/16. Усл. печ. л. 7. Тираж 70 экз.

... самые разные категории читателей найдут в книге что-то интересное для себя.

... будущим слушателям соответствующих курсов (в различных высших учебных заведениях) она очень пригодится, поскольку заполняет существенный пробел в учебной литературе на русском языке.

Александр Шень