ФГОС

10



К. Ю. Поляков
Е. А. Еремин

ИНФОРМАТИКА

1

УГЛУБЛЕННЫЙ УРОВЕНЬ

ИЗДАТЕЛЬСТВО БИНОМ

ФГОС

К. Ю. Поляков, Е. А. Еремин

ИНФОРМАТИКА

УГЛУБЛЕННЫЙ УРОВЕНЬ

Учебник для 10 класса

в 2-х частях

Часть 1

Рекомендовано
Министерством образования и науки
Российской Федерации
к использованию в образовательном процессе
в имеющих государственную аккредитацию
и реализующих образовательные программы
общего образования образовательных учреждениях



Москва
БИНОМ. Лаборатория знаний
2013

УДК 004.9
ББК 32.97
П54

Поляков К. Ю.

П54 Информатика. Углублённый уровень : учебник для 10 класса : в 2 ч. Ч. 1 / К. Ю. Поляков, Е. А. Еремин. — М. : БИНОМ. Лаборатория знаний, 2013. — 344 с. : ил.

ISBN 978-5-9963-1416-4 (Ч. 1)
ISBN 978-5-9963-1152-1

Учебник предназначен для изучения курса информатики на углублённом уровне в 10 классах общеобразовательных учреждений. Содержание учебника опирается на изученный в 7–9 классах курс информатики для основной школы.

Рассматриваются теоретические основы информатики, аппаратное и программное обеспечение компьютера, компьютерные сети, алгоритмизация и программирование, информационная безопасность.

Учебник входит в учебно-методический комплект (УМК), включающий также учебник для 11 класса и компьютерный практикум.

Предполагается широкое использование ресурсов портала Федерального центра электронных образовательных ресурсов (http://fcior.edu.ru/).

Соответствует федеральному государственному образовательному стандарту среднего (полного) общего образования (2012 г.).

УДК 004.9
ББК 32.97

Учебное издание

Поляков Константин Юрьевич, **Еремин** Евгений Александрович

ИНФОРМАТИКА.
УГЛУБЛЁННЫЙ УРОВЕНЬ
Учебник для 10 класса

В двух частях
Часть первая

Ведущий редактор *О. Полежаева*
Ведущие методисты *И. Сретенская, И. Хлобыстова*
Художественное оформление: *И. Марев*
Художественный редактор *Н. Новак*
Иллюстрации: *Я. Соловцова, Ю. Белаш*
Технический редактор *Е. Денюкова*. Корректор *Е. Клитина*
Компьютерная верстка: *Л. Катуркина*

Подписано в печать 26.02.13. Формат 70×100/16.
Усл. печ. л. 27,95. Тираж 10 000 экз. Заказ 48.

Издательство «БИНОМ. Лаборатория знаний»
125167, Москва, проезд Аэропорта, д. 3
Телефон: (499) 157-5272, e-mail: binom@Lbz.ru
http://www.Lbz.ru, http://e-umk.Lbz.ru, http://metodist.Lbz.ru
Отпечатано в ООО ПФ «Полиграфист»,
160001, г. Вологда, ул. Челюскинцев, 3.
Тел.: 8(817-2) 72-61-75; 8(817-2) 72-60-63.

ISBN 978-5-9963-1416-4 (Ч. 1)
ISBN 978-5-9963-1152-1

© БИНОМ. Лаборатория знаний, 2013

Оглавление

От авторов 5

Навигационные значки 8

Глава 1. Информация и информационные процессы 9

§ 1. Информатика и информация 9
§ 2. Что можно делать с информацией? 19
§ 3. Измерение информации 25
§ 4. Структура информации 31

Глава 2. Кодирование информации 55

§ 5. Язык и алфавит 55
§ 6. Кодирование 60
§ 7. Дискретность 76
§ 8. Алфавитный подход к измерению количества информации 84
§ 9. Системы счисления 88
§ 10. Позиционные системы счисления 91
§ 11. Двоичная система счисления 102
§ 12. Восьмеричная система счисления 109
§ 13. Шестнадцатеричная система счисления 114
§ 14. Другие системы счисления 118
§ 15. Кодирование символов 122
§ 16. Кодирование графической информации 127
§ 17. Кодирование звуковой и видеоинформации 146

Глава 3. Логические основы компьютеров 159

§ 18. Логика и компьютер 159
§ 19. Логические операции 161
§ 20. Диаграммы Венна 180
§ 21. Упрощение логических выражений 185
§ 22. Синтез логических выражений 192
§ 23. Предикаты и кванторы 196
§ 24. Логические элементы компьютера 201
§ 25. Логические задачи 210

Глава 4. Компьютерная арифметика 221

§ 26. Особенности представления чисел в компьютере . . . 221
§ 27. Хранение в памяти целых чисел 227
§ 28. Операции с целыми числами 235
§ 29. Хранение в памяти вещественных чисел 252
§ 30. Операции с вещественными числами 261

Глава 5. Устройство компьютера 266

§ 31. История развития вычислительной техники 267
§ 32. Принципы устройства компьютеров 283
§ 33. Магистрально-модульная организация компьютера 294
§ 34. Процессор 301
§ 35. Память 309
§ 36. Устройства ввода 324
§ 37. Устройства вывода 333

От авторов

Вы держите в руках учебник информатики углублённого уровня для 10 класса. Если посмотреть на оглавление, может показаться, что многое из представленного в учебнике материала вам уже знакомо. Действительно, в 7–9 классах вы знакомились с понятиями «информация» и «информационный процесс», изучали кодирование данных, программное обеспечение и основы программирования.

Однако углублённый уровень изучения предполагает, что вы уже начинаете готовиться к освоению будущей профессии. Для этого необходимо более глубокое понимание всех этих вопросов, выход на следующий уровень владения материалом, когда человек может не только воспроизводить полученные знания, но и решать новые сложные задачи с их помощью. Цель этого учебника — дать такие знания, которые позволят вам грамотно решать задачи, не рассмотренные в самом учебнике.

Первые две главы учебника — базовые, в них содержится информация, необходимая для понимания всех последующих глав. Большинство остальных глав можно изучать в разном порядке, они относительно независимы друг от друга.

В углублённом курсе информатики самое серьёзное внимание уделяется разделу «Алгоритмизация и программирование». Авторы выбрали два языка программирования, которые изначально задумывались как учебные языки:

- алгоритмический язык свободно распространяемой системы КуМир (http://lpm.org.ru/kumir2/, http://kpolyakov.narod.ru/school/kumir.htm);
- язык Паскаль, который изучается во многих школах России (http://petriv.ho.com.ua/algo/rus/index.php, http://pascalabc.net/, http://www.freepascal.org/).

Вы можете использовать любой из этих языков, потому что практически все программы и примеры представлены в двух вариантах.

В конце каждого параграфа есть контрольные вопросы, которые помогут понять, хорошо ли вы разобрались в материале. В тексте нет прямых ответов на некоторые вопросы, но есть вся необходимая информация для ответа на них.

Задачи в конце параграфов помогут закрепить материал на практических работах. Самые сложные задачи (на взгляд авторов) отмечены звёздочкой (*).

Мы старались сделать так, чтобы содержание учебника как можно меньше зависело от программного обеспечения, установленного на ваших компьютерах. Весь курс можно успешно изучать, используя только свободное программное обеспечение (СПО) — операционную систему *Linux*, офисный пакет *OpenOffice.org* или его модификации (например, *LibreOffice*), компилятор *FreePascal* и др.

В заключение нам хочется поблагодарить наших коллег, которые взяли на себя труд прочитать предварительные версии отдельных глав учебника и высказать множество полезных замечаний, позволивших сделать учебник более точным, ясным и понятным:

- *А. П. Шестакова*, кандидата педагогических наук, заведующего кафедрой информатики и вычислительной техники Пермского государственного педагогического университета, который вдохновил авторов на написание этого учебника;
- *М. А. Ройтберга*, доктора физико-математических наук, заведующего лабораторией прикладной математики Института математических проблем биологии РАН, г. Пущино;
- *С. С. Михалковича*, кандидата физико-математических наук, доцента кафедры алгебры и дискретной математики Южного федерального университета, г. Ростов-на-Дону;
- *О. А. Тузову*, учителя информатики школы № 550, г. Санкт-Петербург;
- *А. Г. Тамаревскую*, учителя информатики лицея № 533, г. Санкт-Петербург;
- *Н. Д. Шумилину*, кандидата педагогических наук, учителя информатики МОУ «Тверская гимназия № 6», г. Тверь;

- *Л. Б. Кулагину*, учителя информатики ФМЛ № 239, г. Санкт-Петербург;
- *В. Н. Разумова*, учителя информатики МОУ «Большеелховская средняя общеобразовательная школа», с. Большая Елховка, республика Мордовия;
- *Ю. М. Розенфарба*, учителя информатики МОУ «Межозёрная средняя образовательная школа», Челябинская область;
- *Т. А. Мисаренкова*, учителя информатики школы № 163, г. Санкт-Петербург;
- *В. В. Потопахина*, педагога дополнительного образования Краевого центра технического творчества, г. Хабаровск;
- *К. А. Малеванова*, технического директора компании «ПиН Телеком».

Навигационные значки

Уважаемые ученики! В работе с книгой вам помогут навигационные значки:



— важное утверждение или определение.



— вопросы и задания к параграфу.



— дополнительное разъяснение.



— задания для подготовки к итоговой аттестации.



— к каждой главе учебника рекомендуются:

1) электронные образовательные ресурсы (ЭОР) с сайта Федерального центра образовательных ресурсов (ФЦИОР):
   http://fcior.edu.ru
   Доступ к ЭОР из каталога ФЦИОР:
   http://fcior.edu.ru/catalog/meta/4/mc/discipline%2000/mi/4.06/p/page.html.
   Ресурсы размещены в алфавитном порядке, согласно названиям учебных тем;
2) практические работы на методическом сайте издательства lbz.metodist.ru в авторской мастерской К. Ю. Полякова и Е. А. Еремина.



— Проектное или исследовательское задание.
В ходе выполнения проекта (исследования) вы можете:

- подготовить набор полезных ссылок с использованием веб-ресурсов;
- подготовить небольшое выступление с использованием презентации (5–7 мин.);
- оформить доклад и поместить его на сайт школьной конференции;
- подтвердить полученные результаты расчётами и графиками (диаграммами);
- подготовить видеоролик;
- разместить материалы проекта (исследования) в коллекции обучающих модулей по предмету на сайте школы.

Глава 1
Информация и информационные процессы

§ 1
Информатика и информация

Информатика

Задачи, связанные с хранением, передачей и обработкой информации, человеку приходилось решать во все времена: требовалось передавать знания из поколения в поколения, искать нужные книги в хранилищах, шифровать секретную переписку. К концу XIX века количество документов в библиотеках стало настолько велико, что возникла необходимость применить научный подход к задачам хранения и поиска накопленной информации. В это время зародилось новое научное направление, в котором изучалась *документальная* информация, т. е. информация в виде документов (книг, журналов, статей и т. п.). В английском языке оно получило название *information science* (информационная наука, наука об информации).

Применение компьютерной техники значительно увеличило возможности людей в области работы с информацией, позволив автоматизировать рутинную работу. Считается, что слово **«информатика»**[1] в современном значении образовано в результате объединения двух слов: «информация» и «автоматика». Таким образом, получается «автоматическая работа с информацией». В английском языке существует близкое по значению выражение *computer science* (наука о компьютерах).

Современная информатика, которая стала самостоятельной наукой в 70-х годах XX века, изучает теорию и практику обработ-

[1] Впервые этот термин использовал немецкий ученый К. Штейнбух в 1957 г. (нем. *informatik*). В 1962 г. Ф. Дрейфус ввёл слово *informatique* во французский язык, затем оно было переведено на английский (англ. *informatics*).

ки информации с помощью компьютерных систем. Обычно к информатике относят следующие научные направления:

- **теоретическую информатику** (теорию информации, теорию кодирования, математическую логику, теорию автоматов и др.);
- **вычислительную технику** (устройство компьютеров и компьютерных сетей);
- **алгоритмизацию и программирование** (создание алгоритмов и программ);
- **прикладную информатику** (персональные компьютеры, прикладные программы, информационные системы и т. д.);
- **искусственный интеллект** (распознавание образов, понимание речи, машинный перевод, логические выводы, алгоритмы самообучения).

Раньше эти вопросы частично рассматривались в других науках — математике, лингвистике (науке о языке), электронике и др. С появлением компьютеров стало ясно, что все эти направления тесно связаны, и постепенно начала формироваться новая область научной деятельности. Информатика — это область науки в процессе становления, и круг её вопросов в будущем может измениться.

В нашем курсе мы также познакомимся с *информационными технологиями*, которые связаны с применением компьютеров во всех областях современной жизни: при оформлении документов; при подготовке книг и журналов к печати; для расчёта зарплаты; для продажи билетов на поезда и самолёты; для автоматизации производства; при проектировании зданий, кораблей, станков и т. д. Во всех этих сферах используется понятие «*информация*».

Что такое информация?

Латинское слово *informatio* переводится как «*разъяснение*», «*сведения*». В быту под информацией мы обычно понимаем любые сведения или данные об окружающем нас мире и о нас самих. Однако дать общее определение информации весьма непросто. Более того, в каждой области знаний слово «**информация**» имеет свой смысл.

Философы говорят о том, что информация, как зеркало, отражает мир (реальный или вымышленный). Биологи рассматривают информационные процессы в живой природе. Социологи изучают ценность и полезность информации в человеческом обществе. Спе-

циалистов по компьютерной технике в первую очередь интересует представление информации в виде знаков.

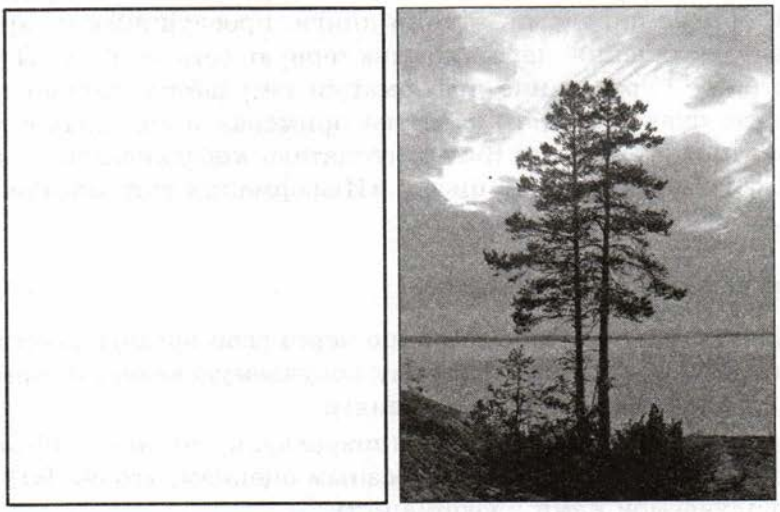
Попробуем посмотреть на информацию с разных сторон и попытаться выявить некоторые её свойства.

Прежде всего информация «бестелесна», или *нематериальна*, она не имеет формы, размеров, массы. С этой точки зрения информация — это то содержание, которое человек с помощью своего сознания «выделяет» из окружающей среды.

Информация **нематериальна**.

Давайте сравним два изображения одинакового размера (рис. 1.1). На первом из них пусто, а на втором мы видим фотографию. Вряд ли кто-то способен долго разглядывать чистый лист, а на фотографию можно долго смотреть, открывая всё новые и новые детали. Почему так?

Первый рисунок разглядывать неинтересно, там всё одинаково — везде белый цвет. На втором рисунке есть *разнообразие*, он неоднороден. Поэтому можно сказать, что он содержит больше информации, чем первый.



1 2

Рис. 1.1

Информация **характеризует разнообразие** (неоднородность) в окружающем мире.

Зачем вообще нам нужна информация? Дело в том, что наше знание всегда в чём-то неполно, в нём есть *неопределённость*. Например, вы стоите на остановке и не знаете, на каком именно автобусе вам нужно ехать в гости к другу (его адрес известен). Неопределённость мешает вам решить свою задачу. Нужный номер автобуса можно определить, например, по карте с маршрутами транспорта. Очевидно, что при этом вы получите новую информацию, которая увеличит знание и уменьшит неопределённость.

При получении информации **уменьшается неопределённость знания**.

Многие выдающиеся ученые XX века (Н. Винер, У. Эшби, К. Шеннон, А. Урсул, А. Моль, В. М. Глушков) давали свое определение информации, но ни одно из них не стало общепринятым. Дело в том, что слово «информация» используется в самых разных ситуациях для обозначения того общего, что есть в разговоре людей, обмене письмами, чтении книги, прослушивании музыки, передаче сообщения через компьютерную сеть и т. д. Поэтому дать строгое определение информации не удаётся, можно только объяснить значение этого слова на примерах и сравнить с другими понятиями. Норберт Винер, создатель *кибернетики* — науки об управлении и связи — писал: «Информация есть информация, а не материя и не энергия».

Как получают информацию

Человек получает информацию через свои органы чувств: глаза, уши, рот, нос и кожу. Поэтому получаемую нами информацию можно разделить на следующие виды:

- *зрительная информация* (визуальная, от англ. *visual*) — поступает через глаза (по разным оценкам, это 80–90% всей получаемой нами информации);
- *звуковая информация* (аудиальная, от англ. *audio*) — поступает через уши;

- *вкусовая информация* — поступает через язык;
- *обонятельная информация* (запахи) — поступает через нос;
- *тактильная информация* — мы её получаем с помощью осязания (кожи), «на ощупь».

Ещё выделяют информацию, получаемую с помощью «мышечного чувства» (человеческий мозг получает импульсы от мышц и суставов при перемещении частей тела).

Некоторые животные чувствуют магнитное поле Земли и используют его для выбора направления движения.

Формы представления информации

Информация может быть представлена (зафиксирована, закодирована) в различных *формах*:

- *текстовая информация* — последовательность символов (букв, цифр, других знаков); в тексте важен порядок их расположения, например КОТ и ТОК — два разных текста, хотя они состоят из одинаковых символов;
- *числовая информация*;
- *графическая информация* (рисунки, картины, чертежи, карты, схемы, фотографии и т. п.);
- *звуковая информация* (звучание голоса, мелодии, шум, стук, шорох и т. п.);
- *мультимедийная информация*, которая объединяет несколько форм представления информации (например, видеоинформация).

Обратим внимание, что одна и та же информация может быть представлена по-разному. Например, результаты измерения температуры в течение недели можно сохранить в виде текста, чисел, таблицы, графика, диаграммы, видеофильма и т. д.

Человек, информация, знания

Обо всех изменениях в окружающем мире человек узнает с помощью своих органов чувств: сигналы от них («первичная» информация) постоянно поступают в мозг. Чтобы понять эти сигналы, т. е. извлечь информацию, человек использует **знания** — свои представления о природе, обществе, самом себе. Знания позволяют человеку принимать решения, определяют его поведение и отношения с другими людьми.

Можно считать, что знания — это модель мира, которая есть у человека. Получив информацию («поняв» сигналы, поступившие от органов чувств), он корректирует эту модель, дополняет свои знания.

Всегда ли полученная информация увеличивает наши знания? Очевидно, что нет. Например, информация о том, что $2 \cdot 2 = 4$ вряд ли увеличит ваши знания, потому что вы это уже знаете, эта информация для вас не нова. Однако она будет новой для тех, кто изучает таблицу умножения. Это значит, что изменение знаний при получении сообщения зависит от того, что человек знал до этого момента. Если он знает всё, что было в полученном сообщении, знания не изменяются.

Вместе с тем сообщение «Учёт вибронных взаимодействий континуализирует моделирование диссипативных структур» (или сообщение на неизвестном языке) также не увеличивает знания, потому что эта фраза, скорее всего, вам непонятна. Иначе говоря, имеющихся знаний не хватает для того, чтобы воспринять новую информацию.

Эти идеи послужили основой *семантической* (смысловой) теории информации, предложенной в 60-х годах XX века советским математиком Ю. А. Шрейдером. На рисунке 1.2 показано, как зависит количество полученных знаний I от того, какая доля информации θ в сообщении уже известна получателю.

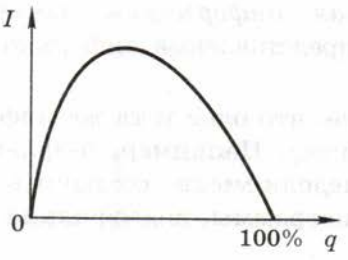


Рис. 1.2

Сообщение увеличивает знания человека, если оно понятно и содержит новые сведения.

К сожалению, «измерить» смысл информации, оценить его числом, довольно сложно. Поэтому для оценки количества инфор-

мации используют другие подходы, о которых вы узнаете чуть позже.

Когда человек хочет поделиться с кем-то своим знанием, он может сказать: «Я знаю, что...» или «Я знаю, как...». Это говорит о том, что есть два разных вида знаний. В первом случае знания — это некоторый известный факт, например: «Я знаю, что Луна вращается вокруг Земли». Такие знания называются *декларативными*, человек выражает их словами (*декларирует*). Декларативные знания — это факты, законы, принципы.

Второй тип знаний («Я знаю, как...») называют *процедурными*. Они выражаются в том, что человек знает, как нужно действовать в той или иной ситуации. К процедурным знаниям относятся алгоритмы решения различных задач.

Для того чтобы сохранить знания и передать другим людям, нужно выразить их на каком-то языке (например, рассказать, записать, нарисовать и т. п.). Только после этого их можно хранить, обрабатывать, передавать, причём с этим может справиться и компьютер. В научной литературе информацию, зафиксированную (закодированную) в какой-то форме, называют **данными**, имея в виду, что компьютер может выполнять с ними какие-то операции, но не способен понимать смысл.

Для того чтобы данные стали **информацией**, их нужно понять и осмыслить, а на это способен только человек. Если человек, получающий сообщение, знает язык, на котором оно записано, он может понять смысл этого сообщения, т. е. получить информацию. Обрабатывая и упорядочивая информацию, человек выявляет закономерности — получает **знания**.

Мы увидели, что в науке существуют достаточно тонкие различия между понятиями «данные», «информация», «знания». Тем не менее на практике чаще всего всё это называется общим термином «информация».

Свойства информации

В идеале информация должна быть:

- *объективной* (не зависящей от чьего-либо мнения);
- *понятной* для получателя;
- *полезной* (позволяющей получателю решать свои задачи);
- *достоверной* (полученной из надёжного источника);

- *актуальной* (значимой в данный момент);
- *полной* (достаточной для принятия решения).

Конечно, информация не всегда обладает всеми этими свойствами. Информация в сообщении «В стакане мало молока» необъективна (для пессимиста полстакана — это мало, а для оптимиста — много). Сообщение 私は散歩に行った。непонятно для не знающих японский язык (оно означает «Я пошёл гулять», только по-японски).

Полезность информации определяется для каждого человека в конкретной ситуации. Например, информация о том, как древние люди добывали огонь, для большинства городских жителей бесполезна, поскольку она никак не помогает им решать свои жизненные проблемы. Вместе с тем в экстремальной ситуации, когда человек оказывается один на один с природой, такие знания очень полезны, потому что сильно увеличивают шансы на выживание, т. е. помогают достичь цели.

Слухи, байки, искажённая информация (в том числе дезинформация) — это примеры недостоверной информации.

Сообщение «10 лет назад здесь был ларёк с мороженым» неактуально, эта информация устарела.

Информация в сообщении «Сегодня будет концерт» неполна, потому что не указано время и участники концерта, и из-за этого мы не можем принять решение (идти или не идти?).

Развитие глобальной сети Интернет, в которую ежеминутно вносится огромное количество самых разнообразных данных, во многом перевернуло привычные представления о работе с информацией. Например, основным источником для поиска учебных материалов теперь фактически является Интернет, а не библиотеки. Однако при использовании информации из Интернета необходимо относиться к ней критически, так как её достоверность никто не гарантирует.

Роль информации в человеческом обществе очень велика. Информация, получаемая нами из разных источников, позволяет принимать решения и во многом определяет всю нашу жизнь. Огромно влияние на общество средств массовой информации (СМИ) — газет, телевидения, изданий в Интернете.

В будущем ожидается переход к информационному обществу, где бóльшая часть населения будет заниматься сбором, обработ-

кой и распространением информации, поэтому высказывание немецкого банкира Н. Ротшильда «Кто владеет информацией, тот владеет миром» становится актуальным как никогда.

Информация в технике

Практически все современные технические устройства (телевизоры, телефоны, стиральные машины, системы управления самолётами и судами и т. д.) строятся на **микропроцессорах**, которые обрабатывают информацию: анализируют сигналы с датчиков, выбирают нужный режим работы. Широко используются **системы программного управления**, например станки, обрабатывающие детали по программе, заложенной в памяти. Эту программу очень легко поменять и настроить станок на изготовление другой детали.

Многие опасные, тяжёлые и утомительные работы за человека могут выполнить **роботы**, у которых датчики заменяют органы чувств. Например, человекоподобный робот (*андроид*) Asimo (рис. 1.3), разработанный фирмой Honda, умеет распознавать предметы, жесты, звуки, узнавать лица, разговаривать через домофон, передавать данные через Интернет.

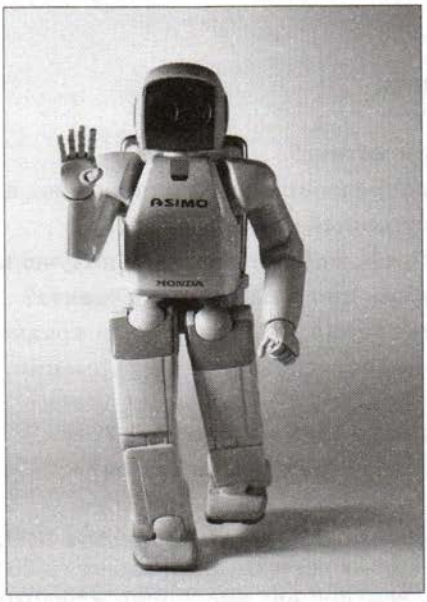


Рис. 1.3. Робот Asimo (www.robotonline.net)

Наиболее универсальным устройством для обработки информации можно считать **компьютер**. Хотя современные компьютеры пока не умеют работать с вкусовой и обонятельной информацией (запахами), работы в этом направлении ведутся. Уже существуют экспериментальные приборы, названные «электронный нос» и «электронный язык»; они построены на основе химических датчиков.

Сейчас в теоретической информатике считается, что компьютер может хранить и обрабатывать только *данные*, но не информацию. Многие учёные полагают, что машина принципиально не может научиться понимать смысл информации и делать выводы[1]. Эту точку зрения подтверждает фактический провал проекта «компьютеров пятого поколения» (Япония, 1980-е гг.), в ходе которого планировалось создать машины, общающиеся с человеком на естественном языке. Тем не менее учёные уделяют этим проблемам огромное внимание. Например, возникло целое научное направление *data mining* («добыча данных»), в котором изучаются методы извлечения информации («смысла», закономерностей, связей, знаний) из огромных наборов данных. В некоторых случаях действительно удаётся использовать огромные вычислительные мощности компьютеров для того, чтобы найти неизвестные ранее закономерности, которые можно использовать на практике.

Вопросы и задания

1. Что изучает информатика?
2. Какие научные направления обычно включают в информатику?
3. Что такое искусственный интеллект?
4. Как связана неопределённость знания с получением информации?
5. Как связана информация и сложность объекта?
6. Объясните, почему термин «информация» трудно определить.
7. Согласны ли вы с «определением» информации, которое дал Н. Винер? Как вы его понимаете?
8. Как человек воспринимает информацию?
9. Чем отличается текст от набора символов?

---

[1] Тем не менее суперкомпьютер Watson фирмы IBM, умеющий отвечать на вопросы, заданные на естественном языке, в 2011 г. выиграл у лучших игроков в телевизионной викторине *Jeopardy!* (аналог телепередачи «Что? Где? Когда?»).

10. К какому виду информации относятся видеофильмы?
11. Что такое тактильная информация?
12. Всякая ли информация увеличивает знания? Почему?
13. На каких идеях основана семантическая теория информации?
14. Приведите примеры своих декларативных и процедурных знаний.
15. В чём, на ваш взгляд, разница между понятиями «данные», «информация», «знания»?
16. Почему считают, что компьютер может работать только с данными?
17. Какими свойствами должна обладать «идеальная» информация?
18. Приведите примеры необъективной, непонятной, бесполезной, недостоверной, неактуальной и неполной информации.
19. Может ли информация быть достоверной, но бесполезной? Достоверной, но необъективной? Объективной, но недостоверной? Актуальной, но непонятной?
20. Приведите примеры обработки информации в технических устройствах.
21. Что умеет робот Asimo? Какую информацию он обрабатывает?
22. Что такое «электронный нос» и «электронный язык»?
23. Как вы считаете, смогут ли компьютеры научиться понимать смысл данных?

Подготовьте сообщение

а) «Информация в жизни общества»
б) «Интернет и изменение уклада жизни людей»
в) «Информационное общество: плюсы и минусы»
г) «Как оценить смысл информации?»

§ 2
Что можно делать с информацией?

Как мы уже знаем, информация сама по себе нематериальна. Поэтому она может существовать только тогда, когда связана с каким-то объектом или средой, т. е. с носителем.

Материальный носитель — это объект или среда, которые могут содержать информацию.

Изменения, происходящие с информацией (т. е. изменения свойств носителя), называются **информационными процессами**. Все эти процессы можно свести к двум основным:

- **передача информации** (данные передаются с одного носителя на другой);
- **обработка информации** (данные изменяются).

Часто информационными процессами называют также и многие другие операции с информацией (например, копирование, удаление и др.), но они, в конечном счёте, сводятся к двум названным процессам.

Для хранения информации тоже используется какой-то носитель. Однако при этом никаких изменений не происходит, поэтому хранение информации нельзя назвать процессом.

Передача информации

При **передаче информации** всегда есть два объекта — источник и приёмник информации. Эти роли могут меняться, например во время диалога каждый из участников выступает то в роли источника, то в роли приёмника информации.

Информация проходит от **источника** к **приёмнику** через **канал связи**, в котором она должна быть связана с каким-то **материальным носителем** (рис. 1.4). Для передачи информации свойства этого носителя должны изменяться со временем. Например, если включать и выключать лампочку, то можно передавать разную информацию, например, с помощью азбуки Морзе.



Рис. 1.4

При разговоре людей носитель информации — это звуковые волны в воздухе. В компьютерах информация передаётся с помощью электрических сигналов или радиоволн (в беспроводных устройствах). Информация может передаваться с помощью света, лазерного луча, телефонной или почтовой связи, компьютерной сети и др.

Информация поступает по каналу связи в виде сигналов, которые приёмник может обнаружить с помощью своих органов чувств (или датчиков) и «понять» (раскодировать).

Сигнал — это изменение свойств носителя, которое используется для передачи информации.

Примеры сигналов — это изменение частоты и громкости звука, вспышки света, изменение напряжения на контактах и т. п.

Человек может принимать сигналы только с помощью своих органов чувств. Чтобы передавать и принимать информацию, например, с помощью радиоволн, нужны вспомогательные устройства: радиопередатчик, преобразующий звук в радиоволны, и радиоприёмник, выполняющий обратное преобразование. Они позволяют расширить возможности человека.

С помощью одного сигнала (одного изменения) невозможно передать много информации. Поэтому чаще всего используется не одиночный сигнал, а *последовательность сигналов*, которая называется **сообщением**. Важно понимать, что сообщение — это только «оболочка» для передачи информации, а информация — это *содержание* сообщения. Приёмник должен сам «извлечь» (раскодировать) информацию из полученной последовательности сигналов. Можно принять сообщение, но не принять информацию, например, услышав речь на незнакомом языке или перехватив шифровку.

Одна и та же информация может быть передана с помощью сообщений, имеющих разные физические носители (например, через устную речь, с помощью записки или с помощью флажного семафора, который используется на флоте) или с помощью разных сообщений. В то же время одно и то же сообщение может нести разную информацию для разных приёмников. Так фраза «В Сантьяго идёт дождь», переданная в 1973 г. на военных радиочастотах, для сторонников генерала Пиночета послужила сигналом к началу государственного переворота в Чили.

К сожалению, в реальном канале связи всегда действуют помехи: посторонние звуки при разговоре, шумы радиоэфира, электрические и магнитные поля. Помехи могут полностью или частично искажать сообщение, вплоть до полной потери информации (например, телефонные разговоры при перегрузке сети).

Чтобы содержание сообщения, искажённого помехами, можно было восстановить, оно должно быть *избыточным*, т. е. в нём должны быть «лишние» элементы, без которых смысл всё равно восстанавливается. Например, в сообщении «Влг впдт в Кспск мр» многие угадают фразу «Волга впадает в Каспийское море», из которой убрали все гласные. Этот пример говорит о том, что естественные языки содержат много «лишнего», их избыточность оценивается в 60–80% (если удалить 60–80% текста, его смысл всё равно удаётся восстановить).

В курсе информатики мы будем рассматривать передачу информации именно как передачу сообщений между компьютерными системами, отвлекаясь от смысла сообщений.

Обработка информации

Обработка — это изменение информации: её формы или содержания. Среди важнейших видов обработки можно назвать:

- *создание новой информации*, например решение задачи с помощью вычислений или логических рассуждений;
- *кодирование* — запись информации с помощью некоторой системы знаков для передачи и хранения; один из вариантов кодирования — шифрование, цель которого — скрыть смысл (содержание) информации от посторонних;
- *поиск информации*, например, в книге, в библиотечном каталоге, на схеме или в Интернете;
- *сортировка* — расстановка элементов списка в заданном порядке, например расстановка чисел по возрастанию или убыванию, расстановка слов по алфавиту; задача сортировки — облегчить поиск и анализ информации.

Для обработки информации человек использует в первую очередь свой мозг. *Нейроны* (нервные клетки) коры головного мозга «переключаются» примерно 200 раз в секунду — значительно медленнее, чем элементы памяти компьютеров. Однако человек практически безошибочно отличает собаку от кошки, а для компьютеров эта задача пока неразрешима. Дело, по-видимому, в том, что мозг решает такие задачи не «в лоб», не путем сложных вычислений, а как-то иначе (как — пока никто до конца не знает).

Компьютер позволяет «усилить» возможности человека в тех задачах обработки информации, решение которых требует длительных расчётов по известным алгоритмам. Однако, в отличие от

человека, для компьютера недоступны фантазия, размышления, творчество.

Хранение информации

Для **хранения** информации человек, прежде всего, использует свою память. Можно считать, что мозг — это одно из самых совершенных хранилищ информации, во многом превосходящее компьютерные средства.

К сожалению, человек многое забывает. Кроме того, необходимо передавать знания другим людям, в том числе и следующим поколениям. Поэтому в древности люди записывали информацию на камне, папирусе, берёсте, пергаменте, затем — на бумаге. В XX веке появились новые средства хранения информации: перфокарты и перфоленты, магнитные ленты и магнитные диски, оптические диски, флеш-память.

В любом случае информация хранится на каком-то **носителе**, который обладает «памятью», т. е. может находиться в разных состояниях, переходить из одного состояния в другое при каком-то внешнем воздействии, и сохранять своё состояние.

При записи информации свойства носителя меняются: на бумагу наносятся текст и рисунки; на магнитных дисках и лентах намагничиваются отдельные участки; на оптических дисках образуются области, по-разному отражающие свет. При хранении эти свойства остаются неизменными, что позволяет потом читать (получать) записанную информацию.

Отметим, что процессы записи и чтения — это процессы передачи информации.

Вопросы и задания



1. Кто (что) может быть источником (приёмником) информации? Приведите примеры.
2. Что такое сигнал? Приведите примеры сигналов.
3. Что такое сообщение? Чем отличается получение информации от получения сообщения?
4. Приведите примеры, когда приём сообщения не означает приём информации.
5. Приведите примеры, когда одна и та же информация может быть передана с помощью разных сообщений.

6. Приведите примеры, когда одно и то же сообщение несёт разную информацию для разных людей.
7. Расскажите, как помехи влияют на передачу информации. Приведите примеры.
8. Что такое избыточность? Почему она полезна при передаче информации?
9. Представьте, что придумали язык, в котором нет избыточности. В чём будет его недостаток?
10. Как вы думаете, какой вариант русского языка обладает наибольшей избыточностью: разговорный, литературный, юридический, язык авиадиспетчеров?
11. В каком из перечисленных выше языков наиболее важна помехоустойчивость? За счёт чего она достигается?
12. Какие виды обработки информации вы знаете?
13. При каких видах обработки информации меняется её содержание?
14. При каких видах обработки информации меняется только форма её представления?
15. К какому виду обработки можно отнести шифрование? Почему?
16. Работники удалённой метеостанции каждые 3 часа измеряют температуру и влажность воздуха и передают данные по рации в районный метеоцентр. Там эти данные сводят в таблицу и отправляют по электронной почте в Гидрометцентр, где мощные компьютеры составляют прогноз погоды. Выделите здесь процессы, связанные с обработкой и передачей информации.
17. Ученик нашёл в старой книге сведения о населении Москвы в XIX веке, составил таблицу по этом данным, построил диаграмму и сделал доклад на школьной конференции. Выделите здесь процессы, связанные с обработкой и передачей информации.
18. Зачем человек записывает информацию?
19. В чём преимущества и недостатки человеческой памяти по сравнению с компьютерной?

Подготовьте сообщение

а) «Компьютер и человек: кто сильнее?»
б) «Носители информации: вчера, сегодня, завтра»

§ 3
Измерение информации

Любая наука рано или поздно приходит к необходимости как-то измерять то, что она изучает. Измерение информации — это одна из важнейших задач теоретической информатики.

Для человека информация — это, прежде всего, смысл, заключённый в сигналах и данных. Как измерить смысл? На этот вопрос пока нет однозначного ответа.

Вспомним, что компьютеры не могут обрабатывать смысл, они работают только с данными (а не с информацией). При этом возникают чисто практические задачи: определить, сколько места займёт на диске текст, рисунок или видеофильм; сколько времени потребуется на передачу файла по компьютерной сети и т. п. Поэтому чаще всего используется **объёмный подход** к измерению информации. Он заключается в том, что количество информации оценивается просто по числу символов, используемых для её кодирования. С этой точки зрения стихотворение А. С. Пушкина и случайный набор букв могут содержать одинаковое количество информации. Конечно, такой подход не универсален, но он позволяет успешно решать практические задачи, связанные с компьютерной обработкой и хранением данных.

Что такое бит?

Рассмотрим электрическую лампочку, которая может находиться в двух состояниях: «горит» и «не горит». Тогда на вопрос «Горит ли сейчас лампочка» есть два возможных варианта ответа, которые можно обозначить цифрами 1 («горит») и 0 («не горит») (рис. 1.5). Поэтому ответ на этот вопрос (полученная информация) может быть записан как 0 или 1[1].

Цифры 0 и 1 называют **двоичными**, и с этим связано название единицы измерения количества информации — **бит**. Английское слово *bit* — это сокращение от выражения *binary digit* — «двоичная цифра». Впервые слово «бит» в этом значении использовал американский инженер и математик Клод Шеннон в 1948 г.

Бит — это количество информации, которую можно записать (закодировать) с помощью одной двоичной цифры.

[1] Конечно, вместо 0 и 1 можно использовать два любых знака.

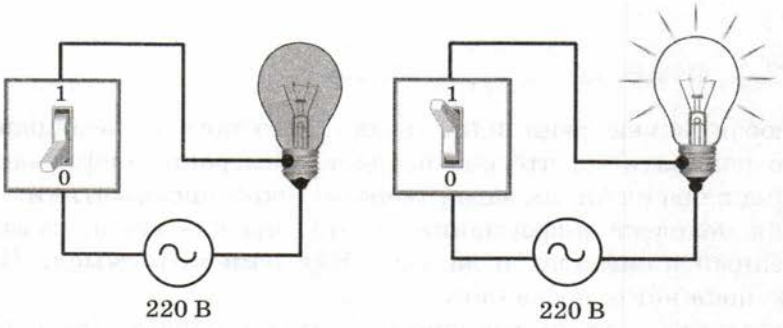


Рис. 1.5

Конечно, нужно договориться, что означают 0 и 1 (1 — это «горит» или «не горит»?), но для измерения количества информации это не важно.

Например, в сообщении «подброшенная монета упала гербом» содержится 1 бит информации, потому что монета могла упасть гербом (обозначим это через 0) или «решкой» (1). Сообщение «Дверь открыта» тоже содержит 1 бит, если считать, что дверь может быть в двух состояниях: открыта (0) или закрыта (1). Вот ещё пример диалога, в котором получена информация в 1 бит:

— Вы будете чай или кофе?

— Кофе, пожалуйста.

2 бита, 3 бита...

А если возможных вариантов не два, а больше? Понятно, что в этом случае количество информации будет больше, чем 1 бит. Представим себе, что на вокзале стоят 4 одинаковых поезда (рис. 1.6), причём только один из них проследует в Москву. Сколько битов понадобится для того, чтобы записать информацию о номере платформы, где стоит поезд на Москву?

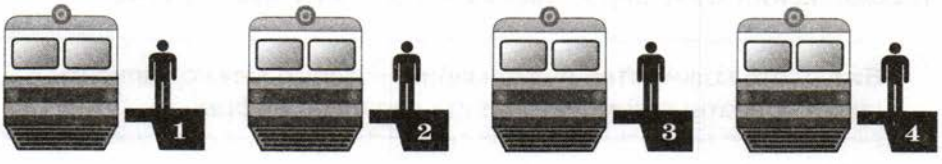


Рис. 1.6

Очевидно, что одного бита недостаточно, так как с помощью одной двоичной цифры можно закодировать только два варианта — коды 0 и 1. А вот два бита дают как раз 4 разных сообщения: 00, 01, 10 и 11. Теперь нужно сопоставить эти коды номерам платформ, например, так: 1 — 00, 2 — 01, 3 — 10, 4 — 11. Тогда сообщение 10 говорит о том, что поезд на Москву стоит на платформе № 3. Это сообщение несёт 2 бита информации.

Три бита дают уже 8 вариантов: 000, 001, 010, 011, 100, 101, 110 и 111. Таким образом, каждый бит, добавленный в сообщение, увеличивает количество вариантов в 2 раза (табл. 1.1).

Таблица 1.1

I, битов	1	2	3	4	5	6	7	8	9	10
N, вариантов	2	4	8	16	32	64	128	256	512	1024

Наверно, вы заметили, что все числа в нижней строчке таблицы — это степени числа 2: $N = 2^I$.

Осталось выяснить, чему равно количество информации, если выбор делается, скажем, из 5 возможных вариантов (или из любого количества, не являющегося степенью числа 2). С точки зрения приведённого выше рассуждения случаи выбора из 5, 6, 7 и 8 вариантов не различаются — для кодирования двух двоичных цифр мало, а трёх — достаточно. Поэтому использование трёх битов для кодирования одного из 5 возможных вариантов *избыточно*, ведь три бита позволяют закодировать целых 8 вариантов! Значит, выбор из 5 вариантов даёт меньше трёх битов информации.

Чтобы количественно измерить разницу между, скажем, 5 и 8 вариантами, придется допустить, что количество информации в битах может быть дробным числом. При этом информация, полученная при выборе из 5 вариантов, больше, чем 2 бита, но меньше, чем 3 бита. Точную формулу для ее вычисления получил в 1928 г. американский инженер Ральф Хартли. Эта формула использует понятие логарифма, поэтому мы познакомимся с ней в 11 классе.

Тем не менее уже сейчас вы можете оценить количество информации при выборе из 5 вариантов. Допустим, на завтрак в лагере отдыха каждый день дают кашу одного из 5 видов. Чтобы закодировать вид каши, которую дают в понедельник, нужно, как мы знаем, 3 бита. Но меню на два дня может быть составлено 25 разными способами (5·5), поэтому для его кодирования достаточно 5 битов, ведь $25 < 32 = 2^5$! Тогда получается, что количество информации при выборе информации из 5 вариантов меньше, чем 5/2 = 2,5 бита. Но и эту оценку можно уточнить. Для трёх дней получаем 5·5·5 = 125 вариантов. Так как $125 < 128 = 2^7$, количество информации при выборе из 5 вариантов составляет не больше, чем 7/3 = 2,33 бита. И так далее. Попробуйте самостоятельно показать, что при выборе из 5 вариантов количество информации больше 2,25 бита. Верно ли, что при выборе из 6 вариантов количество информации менее 2,5 бита?

Другие единицы

Считать большие объёмы информации в битах неудобно хотя бы потому, что придётся работать с очень большими числами (миллиардами, триллионами и т. д.). Поэтому стоит ввести более крупные единицы.

1 байт = 8 битов.

Сразу возникает вопрос: а почему не 10 битов? Дело в том, что слово «байт[1]» (англ. *byte*) имеет второе значение — так называют наименьший блок (ячейку) памяти, который процессор компьютера может считать и обработать за один раз. Для современных компьютеров он состоит из 8 элементов, каждый из которых хранит 1 бит данных. Это связано с тем, что до недавнего времени при обработке текста использовался набор из 256 символов, так что для кодирования каждого символа было нужно 8 битов.

Объёмы данных, с которыми работают компьютеры, нередко измеряются миллионами и миллиардами байтов. В таких случаях используют единицы, образованные с помощью приставок:

1 Кбайт (килобайт) = 1024 байта = 2^{10} байта = 2^{13} битов.
1 Мбайт (мегабайт) = 1024 Кбайт = 2^{10} Кбайт = 2^{20} байтов = 2^{23} битов.
1 Гбайт (гигабайт) = 1024 Мбайт.
1 Тбайт (терабайт) = 1024 Гбайт.

[1] Впервые его использовал американский инженер В. Бухгольц в 1956 г.

Так сложилось исторически, что при измерении количества информации приставка «кило-» обозначает, в отличие от международной системы единиц СИ, увеличение не в 1000 раз, а в $1024 = 2^{10}$ раз. Аналогично «мега-» — это увеличение в $1024^2 = 2^{20} = 1\,048\,576$ раз, а не в 1 млн $= 1000^2$ раз.

Строго говоря, нужно называть такие кило- (мега-, гига-, ...) байты *двоичными*, поскольку множитель 1024 — это 2^{10}. Стандарт Международной электротехнической комиссии (МЭК) предлагает называть их «кибибайт», «мебибайт», «гибибайт» и «тебибайт», но эти названия на практике не прижились.

Для перевода количества информации из одних единиц в другие нужно использовать приведённые выше соотношения. При переводе из крупных единиц в мелкие числа умножают на соотношение между единицами. Например:

$$2 \text{ Кбайт} = 2 \cdot (1 \text{ Кбайт}) = 2 \cdot 1024 \text{ байтов} = 2048 \text{ байтов} = \\ = 2048 \cdot (1 \text{ байт}) = 2048 \cdot 8 \text{ битов} = 16\,384 \text{ бита}.$$

$$2 \text{ Кбайт} = 2 \cdot 2^{10} \text{ байтов} = 2^{11} \text{ байтов} = 2^{11} \cdot 2^3 \text{ битов} = 2^{14} \text{ битов}.$$

В последней строке все расчёты сделаны через степени числа 2, очень часто так бывает проще.

При переводе количества информации из мелких единиц в крупные нужно делить на соотношение между единицами. Например:

$$8192 \text{ бита} = 8192 \cdot (1/8 \text{ байта}) = 8192 : 8 \text{ байтов} = 1024 \text{ байта} = \\ = 1024 \cdot (1/1024 \text{ Кбайт}) = 1024 : 1024 \text{ Кбайт} = 1 \text{ Кбайт}.$$

$$8192 \text{ бита} = 2^{13} \text{ битов} = 2^{13} \cdot (1/2^3 \text{ байта}) = 2^{10} \text{ байтов} = \\ = 2^{10} \cdot (1/2^{10} \text{ Кбайт}) = 1 \text{ Кбайт}.$$

Вопросы и задания

1. Дайте определение минимальной единицы измерения количества информации.
2. Приведите примеры сообщений, количество информации в которых равно 1 биту.
3. Что такое двоичные цифры?
4. Объясните, почему все числа во второй строке табл. 1.1 — это степени числа 2.
5. Какие единицы используют для измерения больших объёмов информации?
6. Что означают приставки «кило-», «мега-», «гига-» и «тера-» при измерении количества информации?

7. Какие приставки рекомендует МЭК для обозначения двоичных килобайта и мегабайта? Как вы думаете, почему они редко используются?

Задачи

1. Пассажир не знает, какой (только один!) из 8 поездов, стоящих на вокзале, проследует в Санкт-Петербург. В справочном бюро он задаёт 8 вопросов: «Поезд на 1-й платформе проследует в Санкт-Петербург?», «Поезд на 2-й платформе проследует в Санкт-Петербург?» и т. д. На первые 7 вопросов он получает ответ «нет», а на последний — «да». Пассажир считает, что он получил 8 битов информации. Прав он или нет? Почему?

*2. В зоопарке содержится 10 обезьян, причём одна из них выступает в цирке. Обезьяны сидят в двух вольерах, в первом — 8 животных, а во втором — два. Посетитель зоопарка считает, что сообщение «Обезьяна, выступающая в цирке, сидит во втором вольере» содержит 1 бит информации. Прав он или нет? Рассмотрите разные варианты уточнения постановки задачи.

3. В горах, рядом с которыми живёт племя Тумба-Юмба, есть 4 пещеры. В каждой из них может быть (а может не быть) клад. Можно ли закодировать сведения о том, где есть клады, используя 3 бита? 4 бита? 5 битов?

4. Известно, что ровно в двух пещерах из четырёх есть клады. Сколько битов нужно, чтобы закодировать информацию о расположении кладов?

*5. Известно, что дверь с двумя замками открывается двумя из четырёх имеющихся ключей. Оцените количество информации в сообщении «Дверь открывается ключами № 2 и № 4». Закодируйте его, используя наименьшее количество двоичных цифр.

*6. Известно, что дверь открывается двумя из пяти имеющихся ключей. Оцените количество информации в сообщении «Верхний замок открывается ключом № 1, а нижний — ключом № 4». Закодируйте его, используя наименьшее количество двоичных цифр.

7. Вася задумал число от 1 до 100. Нужно отгадать это число за наименьшее число попыток, задавая Васе вопросы, на которые он отвечает только «да» и «нет». За сколько вопросов вы берётесь угадать число? Как нужно задавать вопросы, чтобы их число было минимальным даже в худшем случае?

8. Даниил задумал число от 20 до 83. Сколько битов информации содержится в сообщении «Даниил задумал число 77»? Закодируйте это сообщение, используя наименьшее количество двоичных цифр.

9. Двое играют в «крестики-нолики» на поле размером 4 × 4 клетки. Какое количество информации получил второй игрок, узнав первый ход соперника?
10. На вокзале посёлка Сосново три платформы, у каждой из них стоит поезд. Девушка в справочном окне отвечает на все вопросы только «да» и «нет». За какое минимальное число вопросов можно узнать, в каком порядке отходят поезда?
11. Переведите 1 Мбайт во все изученные единицы измерения количества информации.
12. Переведите 2^{26} битов во все изученные единицы измерения количества информации.
13. Сколько килобайтов содержится в 32 768 битах?
14. Сколько битов в 8 Кбайтах?
15. Сколько битов содержит 1/16 Кбайт?
16. Сколько битов содержит 1/512 Мбайт?

Подготовьте сообщение

а) «Бит и байт: как возникли термины?»
б) «Стандарт МЭК и единицы измерения количества информации»

§ 4
Структура информации

Зачем структурировать информацию?

Давайте сравним четыре сообщения.

Первое:
«Для того чтобы добраться из Москвы до села Васино, нужно сначала долететь на самолёте до города Ивановска. Затем на электричке доехать до Ореховска. Там на пароме переправиться через реку Слоновую в посёлок Ольховка, и оттуда ехать в село Васино на попутной машине».

Второе:
«Как ехать в Васино:
1. На самолёте из Москвы до г. Ивановска.
2. На электричке из г. Ивановска до г. Ореховска.
3. На пароме из г. Ивановска через р. Слоновую в пос. Ольховка.
4. На попутной машине из пос. Ольховка до с. Васино».

Третье:

Откуда	Куда	Транспорт
Москва	г. Ивановск	Самолёт
г. Ивановск	г. Ореховск	Электричка
г. Ореховск	пос. Ольховка	Паром (р. Слоновая)
пос. Ольховка	с. Васино	Попутная машина

Четвёртое (рис. 1.7):

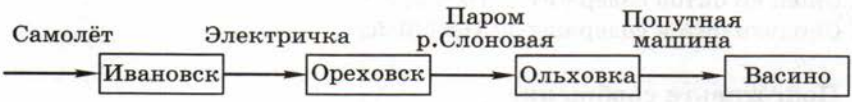


Рис. 1.7

Можно считать, что все эти (такие разные по форме!) сообщения содержат одну и ту же информацию. Какие из них проще воспринимать? Очевидно, что человеку «вытащить» полезную информацию из сплошного текста (первое сообщение) сложнее всего. Во втором случае мы сразу видим все этапы поездки и понимаем, в каком порядке они следуют друг за другом. Третье сообщение (таблицу) и четвёртое (схему) можно понять сразу, с первого взгляда. Второй, третий и четвёртый варианты воспринимаются лучше и быстрее первого, потому что в них выделена **структура** информации, в которой самое главное — этапы поездки в Васино.

Зачем книгу разбивают на главы и разделы, а не пишут сплошной текст? Зачем в тексте выделяют абзацы? Прежде всего для того, чтобы подчеркнуть основные мысли, — каждая глава, раздел, абзац содержат определённую идею. Благодаря особенностям человеческого восприятия, при таком выделении структуры улучшается передача информации от автора к читателю.

При автоматической (компьютерной) обработке правильно выбранная структура данных облегчает доступ к ним, позволяет быстро найти нужные данные.

Средства, облегчающие поиск информации, знакомы вам по работе с книгами. Самый простой (но очень утомительный!) способ найти в книге то, что нужно, — перелистывать страницу за страницей. Однако в большинстве книг есть оглавление, которое

позволяет сразу найти нужный раздел, и это значительно ускоряет поиск.

В словарях слова всегда расставлены в алфавитном порядке (представьте себе, что было бы, если бы они были расположены произвольно!). Поэтому, открыв словарь в любом месте, мы можем сразу определить, куда дальше листать страницы для поиска нужного слова — вперёд или назад.

В больших книгах используют *указатели* (индексы) (рис. 1.8) — списки основных терминов с указанием страниц, на которых они встречаются.

Оглавление:

1. Информация 5
1.1. Что такое информация? 6
1.2. Виды информации 8
1.3. Информация в технике 11
2. Измерение информации 12
2.1. Что такое бит? 13
2.2. Байт и другие единицы 14

Словарь:

автомат — *automaton*
автор — *author*
адрес — *address*
алгебра — *algebra*
алгоритм — *algorithm*
архив — *archive*
архитектура — *architecture*
асимметрия — *asymmetry*

Указатель:

А
аксиома 45
алгоритм 30, 78
архиватор 125
Б
бит 5, 15, 25, 43
брандмауэр 112
браузер 322

Рис. 1.8

Структурирование — это выделение важных элементов в информационных сообщениях и установление связей между ними.

Цели структурирования для человека — облегчение восприятия и поиска информации, выявление закономерностей. При компьютерной обработке структурирование ускоряет поиск нужных данных.

Знакомые структуры данных

С некоторыми структурами данных вы уже знакомы. Например, на уроках математики вы изучали **множество** — некоторый набор элементов. Чтобы определить множество, мы должны перечислить все его элементы (например, множество, состоящее из Васи, Пети и Коли) или определить характерный признак, по

которому элементы включаются в это множество (например, множество драконов с пятью зелёными хвостами или множество точек, в которых функция принимает положительные значения).

Множество может состоять из конечного числа элементов (множество букв русского алфавита), бесконечного числа элементов (множество натуральных чисел) или вообще быть пустым (множество слонов, живущих на Северном полюсе). Множества, с которыми работает компьютер, не могут быть бесконечными, потому что его память конечна.

В документах множество часто оформляют в виде маркированного списка, например:

- процессор;
- память;
- устройства ввода;
- устройства вывода.

В таком списке порядок элементов не важен, от перестановки элементов множество не меняется (рис. 1.9).



Рис. 1.9

Линейный список состоит из конечного числа элементов, которые должны быть расположены в строго определённом порядке. В отличие от множества элементы в списке могут повторяться. Список обычно упорядочен (отсортирован) по какому-то правилу, например по алфавиту, по важности, по последовательности действий и т. д. В тексте он часто оформляется как нумерованный список, например:

1) надеть носки;
2) надеть ботинки;
3) выйти из дома.

Переставить местами элементы такого списка нельзя (это будет уже другой список). Список можно задать перечислением элементов, с первого до последнего:

(надеть носки, надеть ботинки, выйти из дома),

а также представить в виде цепочки связанных элементов (рис. 1.10).

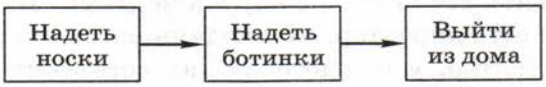


Рис. 1.10

Ещё одна знакомая вам структура — **таблица**. С помощью таблиц устанавливается связь между несколькими элементами. Например, в табл. 1.2 элементы в каждой строке связаны между собой — это свойства некоторого объекта (человека).

Таблица 1.2

Фамилия	Имя	Рост, см	Вес, кг	Год рождения
Иванов	Иван	175	67	1996
Петров	Пётр	164	70	1998
Сидоров	Сидор	168	63	2000

Именно так хранится информация в базах данных: строка таблицы, содержащая информацию об одном объекте, называется *записью*, а столбец (название свойства) — *полем*.

Возможен и другой вариант таблицы, когда роли строк и столбцов меняются. В первом столбце записываются названия свойств, а данные в каждом из следующих столбцов описывают свойства какого-то объекта. Например, табл. 1.3 содержит характеристики разных марок автомашин.

Таблица 1.3

Марка	Лада Приора	Лада Калина	ВАЗ 2110	ВАЗ 21099
Мощность двигателя, л. с.	98	89	79	70
Максимальная скорость, км/ч	183	165	165	156
Время разгона до 100 км/ч, с	11,5	12,5	14	15

Иерархия (дерево)

Линейных списков и таблиц иногда недостаточно для того, чтобы представить все связи между элементами. Например, в некоторой фирме есть директор, ему подчиняются главный инженер и главный бухгалтер, у каждого из них есть свои подчинённые. Если мы захотим нарисовать схему управления этой фирмы, она получится *многоуровневой* (рис. 1.11, *а*).

Такая структура, в которой одни элементы «подчиняются» другим, называется **иерархией** (от древнегреческого ἱεραρχία — священное правление). В информатике иерархическую структуру называют **деревом**. Дело в том, что, если перевернуть схему на рис. 1.11 вверх ногами, она становится похожа на дерево (точнее, на куст, см. рис. 1.11, *б*).



а *б*

Рис. 1.11

Дерево состоит из узлов и связей между ними (они называются **дугами**). Самый первый узел, расположенный на верхнем уровне (в него не входит ни одна стрелка-дуга), — это **корень дерева**. Конечные узлы, из которых не выходит ни одна дуга, называются **листьями**. Все остальные узлы, кроме корня и листьев, — промежуточные.

Из двух связанных узлов тот, который находится на более высоком уровне, называется **родителем**, а другой — **сыном**. Корень — это единственный узел, у которого нет родителя; у листьев нет сыновей.

Используются также понятия **предок** и **потомок**. Потомок какого-то узла — это узел, в который можно перейти по стрелкам от узла-предка. Соответственно, предок какого-то узла — это узел, из которого можно перейти по стрелкам в данный узел.

В дереве на рис. 1.12 родитель узла E — это узел B, а предки узла E — это узлы A и B, для которых узел E — потомок. Потомками узла A (корня) являются все остальные узлы.

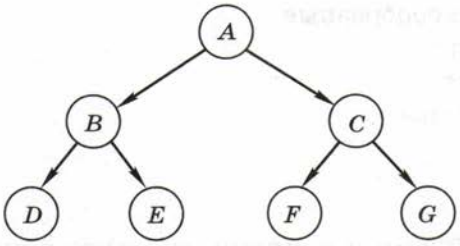


Рис. 1.12

Типичный пример иерархии — различные *классификации* (животных, растений, минералов, химических соединений). Например, отряд *Хищные* делится на два подотряда: *Псообразные* и *Кошкообразные*. В каждом из них выделяют несколько семейств (рис. 1.13).

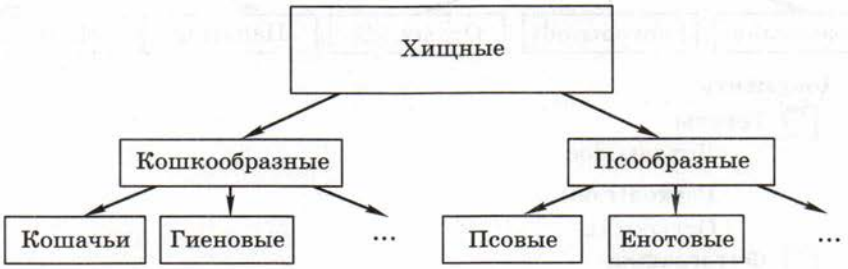


Рис. 1.13

Конечно, на рис. 1.13 показаны не все семейства, остальные обозначены многоточиями.

В текстах иерархию часто представляют в виде многоуровневого списка. Например, оглавление книги о хищниках может выглядеть так:

Глава 1. Псообразные
1.1. Псовые
1.2. Енотовые
1.3. Медвежьи
...

Глава 2. Кошкоообразные
2.1. Кошачьи
2.2. Гиеновые
2.3. Мангустовые
...

Работая с файлами и папками, мы тоже встречаемся с иерархией: классическая файловая система имеет древовидную структуру[1]. Вход в папку — это переход на следующий (более низкий) уровень иерархии (рис. 1.14).

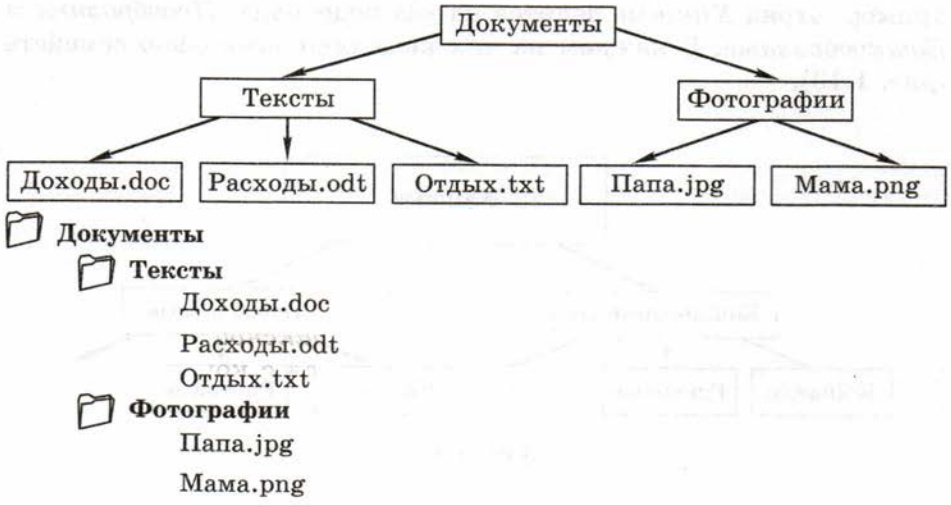


Рис. 1.14

Алгоритм вычисления арифметического выражения тоже может быть представлен в виде дерева (рис. 1.15).

$$(a + 3) * 5 - 2 * b$$

[1] В современных файловых системах (NTFS, ext3) файл может «принадлежать» нескольким каталогам одновременно. При этом древовидная структура, строго говоря, нарушается.

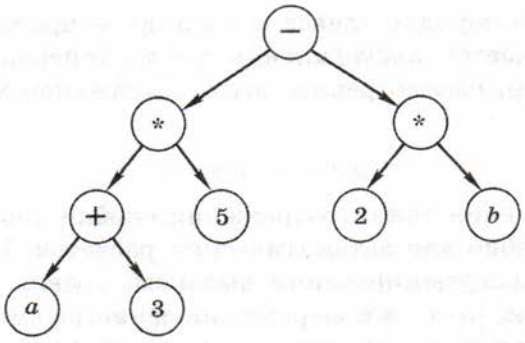


Рис. 1.15

Здесь листья — это числа и переменные, тогда как корень и промежуточные вершины — знаки операций. Вычисления идут «снизу вверх», от листьев — к корню. Показанное дерево можно записать так:

$$(-(*(+(a, 3), 5), *(2, b)))$$

Самое интересное, что скобки здесь необязательны; если их убрать, то выражение всё равно может быть однозначно вычислено:

$$- * + a\ 3\ 5 * 2\ b$$

Такая запись, которая называется *префиксной* (операция записывается перед данными), просматривается с конца. Как только встретится знак операции, эта операция выполняется с двумя значениями, записанными справа. В рассмотренном выражении сначала выполняется умножение:

$$- * + a\ 3\ 5\ \ (2 * b)$$

затем — сложение:

$$- *\ \ (a + 3)\ 5\ \ (2 * b)$$

и ещё одно умножение:

$$-\ (a + 3) * 5\ \ (2 * b)$$

и наконец, вычитание:

$$(a + 3) * 5 - (2 * b)$$

Для получения префиксной записи мы обходили все узлы дерева в порядке «корень — левое поддерево — правое поддерево». Действительно, сначала записана метка корня («−»), затем все метки левого поддерева, а затем — все метки правого поддерева. Для поддеревьев используется тот же порядок обхода. Если же

обойти дерево в порядке «левое поддерево — правое поддерево — корень», получается *постфиксная* форма (операция *после* данных). Например, рассмотренное выше выражение может быть записано в виде

$$a\ 3 + 5 * 2\ b * -$$

Для вычисления такого выражения скобки также не нужны, и это очень удобно для автоматических расчётов. Когда программа на языке программирования высокого уровня переводится в машинные коды, часто все выражения записываются в бесскобочной постфиксной форме и именно так и вычисляются. Постфиксная форма для компьютера удобнее, чем префиксная, потому что, когда программа доходит до знака операции, все данные для выполнения этой операции уже готовы.

Графы

Подумайте, как можно структурировать такую информацию:

«От посёлка Васюки три дороги идут в посёлки Солнцево, Грибное и Ягодное. Между Солнцевым и Грибным и между Грибным и Ягодным также есть дороги. Кроме того, есть дорога, которая идет из Грибного в лес и возвращается обратно в Грибное».

Можно, например, нарисовать схему дорог (рис. 1.16, *а*). На рисунке 1.16, *б* населённые пункты для краткости обозначены латинскими буквами.



а *б*

Рис. 1.16

Для исследования таких схем используют графы.

Граф — это набор вершин и связей между ними (рёбер).

Для хранения информации о вершинах и связях графа, соответствующего схеме на рис. 1.16, можно использовать таблицу (матрицу), показанную на рис. 1.17.

	A	*B*	*C*	*D*
A	0	1	1	0
B	1	0	1	1
C	1	1	1	1
D	0	1	1	0

Рис. 1.17

Единица на пересечении строки *A* и столбца *B* означает, что между вершинами *A* и *B* есть связь. Ноль указывает на то, что связи нет. Такая таблица называется **матрицей смежности**. Она симметрична относительно главной диагонали (серые клетки в таблице).

На пересечении строки *C* и столбца *C* стоит единица, которая говорит о том, что в графе есть **петля** — ребро, которое начинается и заканчивается в одной и той же вершине.

Можно поступить иначе: для каждой вершины перечислить все вершины, с которыми связана данная вершина. В этом случае мы получим **список смежности**. Для рассмотренного графа список смежности выглядит так:

$$(A\,(B, C), B\,(A, C, D), C\,(A, B, C, D), D\,(B, C))$$

Строго говоря, граф — это математический объект, а не рисунок. Конечно, его можно нарисовать на плоскости (например, как на рис. 1.16, *б*), но матрица смежности и список смежности не дают никакой информации о том, как именно следует располагать узлы друг относительно друга. Для таблицы на рис. 1.17 возможны, например, варианты, показанные на рис. 1.18.

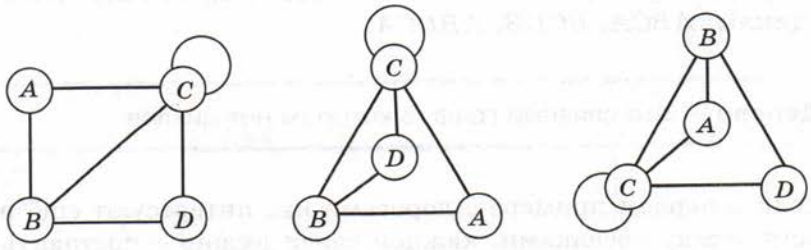


Рис. 1.18

В рассмотренном примере все вершины связаны, т. е. между любой парой вершин существует **путь** — последовательность рёбер, по которым можно перейти из одного узла в другой. Такой граф называется **связным**.

Связный граф — это граф, между любыми вершинами которого существует путь.

Теперь представьте себе, что дороги Васюки — Солнцево, Васюки — Грибное и Грибное — Ягодное завалило снегом (или размыло дождём) так, что по ним не пройти и не проехать (рис. 1.19).

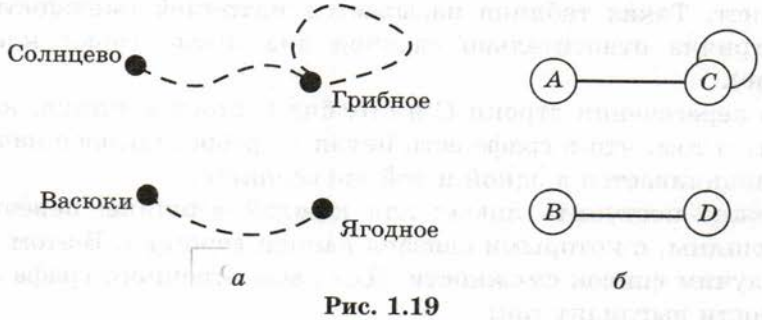


Рис. 1.19

Схему на рис. 1.19, *а* тоже можно считать графом (она подходит под определение), но в таком графе есть две несвязанные части, каждая из которых — связный граф. Такие части называют **компонентами связности**.

Вспоминая материал предыдущего пункта, можно сделать вывод, что дерево — это частный случай связного графа. Но у дерева есть одно важное свойство — в нём нет замкнутых путей (**циклов**). Граф на рис. 1.17 не является деревом, потому что в нём есть циклы: *ABCA*, *BCDB*, *ABDCA*.

Дерево — это связный граф, в котором нет циклов.

Если в первом примере с дорогами нас интересуют ещё и расстояния между поселками, каждой связи нужно сопоставить число (**вес**) (рис. 1.20).

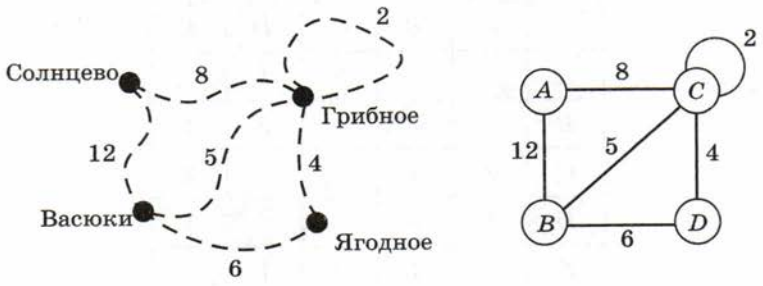


Рис. 1.20

Такой граф называется **взвешенным**, поскольку каждое ребро имеет свой вес. Весом может быть не только расстояние, но и, например, стоимость проезда или другая величина.

Как хранить информацию о таком графе? Ответ напрашивается сам собой — нужно в таблицу записывать не 1 или 0, а вес ребра. Если связи между двумя вершинами нет, на бумаге можно оставить ячейку таблицы пустой, а при хранении в памяти компьютера записывать в неё условный код, например –1. Такая таблица называется **весовой матрицей**, потому что содержит веса рёбер. В данном случае она выглядит, как показано на рис. 1.21.

	A	*B*	*C*	*D*
A		12	8	
B	12		5	6
C	8	5	2	4
D		6	4	

Рис. 1.21

Так же как и матрица смежности, весовая матрица симметрична относительно главной диагонали. Нижняя ячейка в столбце *A* и верхняя в столбце *D* говорят о том, что между вершинами *A* и *D* нет ребра.

Если в графе немного вершин, весовая матрица позволяет легко определить наилучший маршрут из одной вершины в другую простым перебором вариантов. Рассмотрим граф, заданный весовой матрицей на рис. 1.22 (числа определяют стоимость поездки между соседними пунктами).

	A	B	C	D	E
A			3	1	
B			4	5	1
C	3	4			2
D	1	5			1
E		1	2	1	

Рис. 1.22

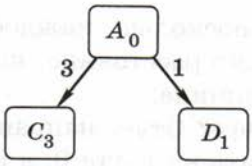


Рис. 1.23

Найдём наилучший путь из A в B — такой, при котором общая стоимость поездки минимальная. Сначала видим, что из пункта A напрямую в B ехать нельзя, а можно ехать только в C и D. Изобразим это на схеме (рис. 1.23).

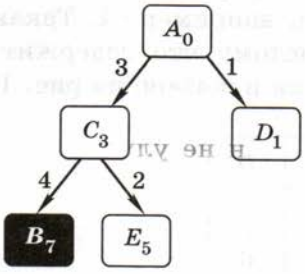


Рис. 1.24

Числа около рёбер показывают стоимость поездки по этому участку, а индексы у названий вершин показывают общую стоимость проезда в данную вершину из вершины A.

Теперь разберём варианты дальнейшего движения из вершины C (вершину A уже не нужно рассматривать, так как мы из неё пришли) (рис. 1.24).

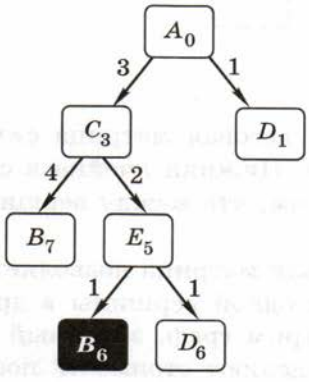


Рис. 1.25

Видим, что из C сразу можно попасть в B, стоимость проезда в этом случае равна 7. Но, возможно, это не самый лучший вариант, и нужно проверить ещё путь через вершину E. Действительно, оказывается, что можно сократить стоимость до 6 (рис. 1.25).

Исследовать дальше маршрут, содержащий цепочку *ACED*, нет смысла, потому что его стоимость явно будет больше 6. Аналогично строим вторую часть схемы (вы догадались, что схема представляет собой дерево!) (рис. 1.26).

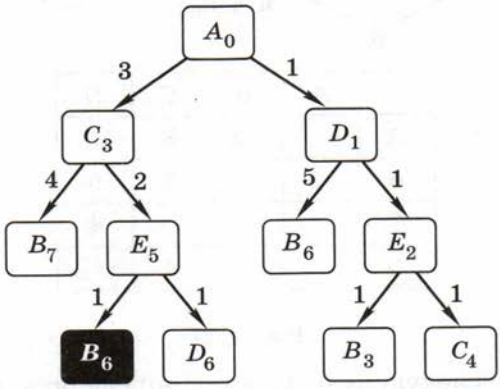


Рис. 1.26

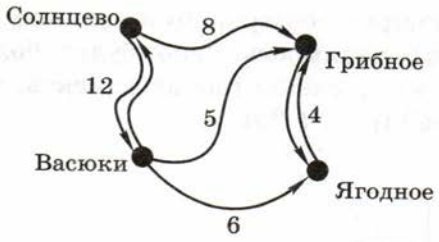
Таким образом, **оптимальный (наилучший) маршрут** — *ADEB*, его стоимость — 3. Маршрут *ADEC*, не дошедший до вершины *B*, далее проверять не нужно, он не улучшит результат.

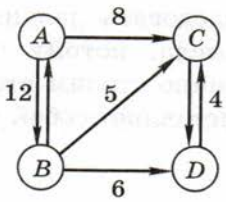
Конечно, для более сложных графов метод перебора работает очень долго, поэтому используются более совершенные (но более сложные) методы, о которых мы поговорим в 11 классе.

Наверное, вы заметили, что при изображении деревьев, которые описывают иерархию (подчинение), мы ставили стрелки от верхних уровней к нижним. Это означает, что для каждого ребра указывается направление, и двигаться можно только по стрелкам, но не наоборот. Такой граф называется **ориентированным** (или коротко **орграфом**). Он может служить, например, моделью системы дорог с односторонним движением. Матрица смежности и весовая матрица для орграфа уже не обязательно будут симметричными.

На схеме на рис. 1.27 всего две дороги с двусторонним движением, по остальным можно ехать только в одну сторону.

Рёбра в орграфе называют **дугами**. Дуга, в отличие от ребра, имеет начало и конец.





	A	B	C	D
A		12	8	
B	12		5	6
C				4
D			4	

Рис. 1.27

Рассмотрим следующую задачу: определить количество возможных путей из вершины *А* в вершину *К* для ориентированного графа, показанного на рис. 1.28.

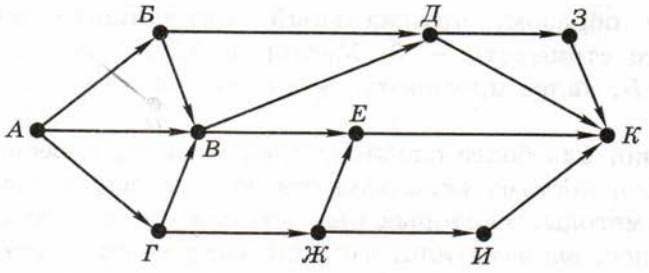


Рис. 1.28

Так как граф ориентированный, переходить в другую вершину можно только по стрелкам.

Будем двигаться по стрелкам от начальной вершины *А*. Около каждой вершины запишем количество возможных путей из вершины *А* в эту вершину. Найдём все вершины, в которые можно попасть только из *А*: это вершины *Б* и *Г*, записываем около них количество путей 1 (рис. 1.29).

Теперь ищем вершины, в которые можно попасть только из уже отмеченных вершин. Например, в вершину *В* есть один путь из *А* напрямую, а также по одному пути через *Б* и *Г* (так как эти вершины отмечены числом 1). Общее количество путей из *А* в *В*

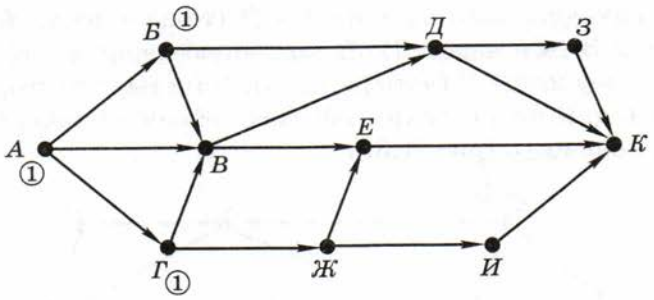


Рис. 1.29

вычисляется как сумма отметок предыдущих вершин. В данном случае получаем всего три маршрута из *А* в *В*. В вершину *Ж* можно попасть только из *Г*, поэтому число путей в *Г* и *Ж* совпадает (рис. 1.30).

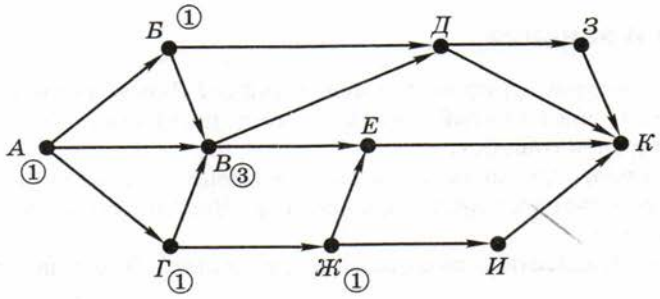


Рис. 1.30

В вершину *Д* идёт один путь через *Б* и три пути через *В*, поэтому общее число путей — четыре. Аналогично получаем четыре пути в вершину *Е*: три пути через *В* и один — через *Ж* (рис. 1.31).

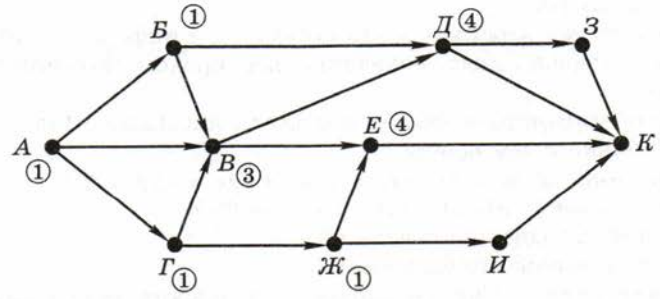


Рис. 1.31

Далее находим один путь из *А* в *И* (только через *Ж*) и четыре пути из *А* в *З* (все через *Д*). В конечную вершину *К* можно попасть через вершины *Д* (четыре пути), *З* (четыре пути), *Е* (четыре пути) и *И* (один путь), таким образом, общее количество различных путей равно 13 (рис. 1.32).

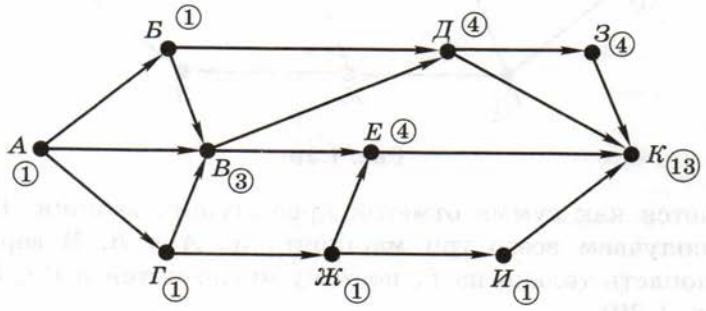


Рис. 1.32

Вопросы и задания

1. Что такое структурирование информации? Зачем оно нужно?
2. Что такое алфавитный порядок? Как поступают, если начальные символы слов совпали?
3. Расскажите, как используются оглавление, словарь и индекс для быстрого поиска нужной информации. Чем эти средства отличаются друг от друга?
4. Какими способами можно задать множество? Что такое пустое множество?
5. Приведите примеры множеств.
6. Чем отличаются множество и линейный список?
7. Что такое матрица?
8. Как можно записать табличные данные в виде списка?
9. Что такое иерархия? Приведите примеры.
10. Вспомните известные вам классификации, которые вы изучали на других предметах.
11. Как называется иерархическая структура в информатике?
12. Что такое корень, лист, родитель, сын, предок, потомок в структуре «дерево»?
13. В дереве 4 потомка, и все они являются листьями. Нарисуйте это дерево. Сколько в нём вершин?
14. В чём разница между понятиями «ребро» и «дуга»?
15. В чём различие понятий «дерево», «граф»?
16. Какой граф называется связным?
17. Что такое компонента связности?
18. Что такое петля? Как по матрице смежности определить, есть ли петли в графе?

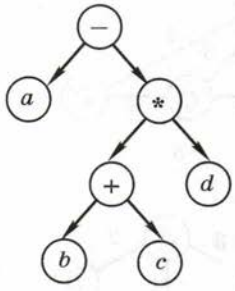
19. Что такое орграф? Когда для представления данных используются орграфы? Приведите примеры.
20. Выберите наиболее подходящий способ структурирования информации для хранения:
    а) данных о крупнейших озерах мира;
    б) рецепта приготовления шашлыка;
    в) схемы железных дорог;
    г) схемы размещения файлов на флэш-диске.

Подготовьте сообщение

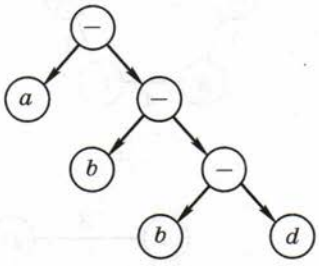
а) «Как вычисляются арифметические выражения?»
б) «Постфиксная и инфиксная формы записи выражений»
в) «Графы в практических задачах»
г) «Диаграммы связей (mind maps)»
д) «Системы классификации книг (ДКД, УДК)»

Задачи

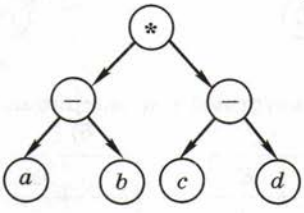
1. Определите выражения, соответствующие каждому из деревьев, в «нормальном» виде со скобками (эту форму называют *инфиксной* — операция записывается между данными). Постройте для каждого из них постфиксную форму.



а



б



в

2. Постройте деревья, соответствующие следующим арифметическим выражениям. Запишите эти выражения в префиксной и постфиксной формах:
    а) $(a+b)*(c+2*d)$
    б) $(2*a-3*d)*c+2*b$
    в) $(a+b+2*c)*d$
    г) $3*a-(2*b+c)*d$

3. Вычислите выражения, записанные в постфиксной форме:

а) 12 6 + 7 3 – 1 – * 12 +

б) 12 10 – 5 7 + * 7 – 2 *

в) 5 6 7 8 9 + – + –

г) 5 4 3 2 1 – – – –

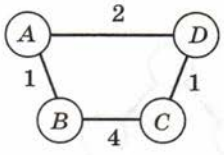
Запишите каждое из них в инфиксной и в префиксной формах и постройте соответствующее дерево. Единственно ли такое дерево? В этом дереве назовите корень, листья и промежуточные вершины.

4. Нарисуйте граф, в котором 5 вершин и три компоненты связности. Постройте его матрицу смежности.

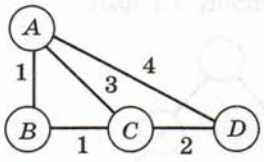
5. Структурируйте следующую информацию разными способами: «Между посёлками Верхний и Нижний есть просёлочная дорога длиной 10 км. Село Сергеево соединяется двумя асфальтовыми шоссе с Нижним (22 км) и Верхним (16 км). В село Солнечное можно доехать только из Сергеева по грунтовой дороге (5 км)». Можно ли сказать точно, как расположены эти пункты?

6. Для графа, полученного в предыдущей задаче, постройте матрицу смежности, список смежности, весовую матрицу. Является ли этот граф деревом?

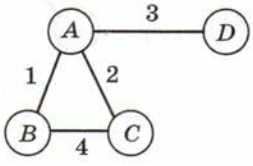
7. Постройте матрицы смежности и весовые матрицы графов:



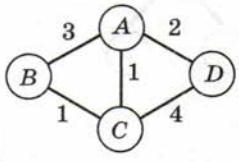
а



б



в



г

8. Постройте графы, соответствующие матрицам смежности.

а)

	A	*B*	*C*	*D*	*E*
A		0	1	1	0
B	0		1	0	1
C	1	1		0	1
D	1	0	0		0
E	0	1	1	0	

б)

	A	*B*	*C*	*D*	*E*
A		0	1	1	1
B	0		1	0	0
C	1	1		0	1
D	1	0	0		0
E	1	0	1	0	

в)

	A	B	C	D	E
A		0	1	1	1
B	0		1	0	1
C	1	1		0	1
D	1	0	0		0
E	1	1	1	0	

г)

	A	B	C	D	E
A		0	0	1	0
B	0		1	0	1
C	0	1		1	1
D	1	0	1		0
E	0	1	1	0	

9. Постройте графы, соответствующие весовым матрицам.

а)

	A	B	C	D	E
A		4	3		7
B	4			2	
C	3			6	
D		2	6		1
E	7			1	

б)

	A	B	C	D	E
A		2	5		6
B	2			3	
C	5				
D		3			1
E	6			1	

в)

	A	B	C	D	E
A			2	2	6
B				2	
C	2			2	
D	2	2	2		
E	6				

г)

	A	B	C	D	E
A		5	2		6
B	5			5	
C	2			2	
D		5	2		3
E	6			3	

10. Стоимость перевозок между пунктами, которые для краткости обозначены буквами A, B, C, D и E, задаётся таблицей (весовой матрицей графа). Нужно перевезти груз из пункта A в пункт B. Для каждого из четырёх вариантов определите оптимальный маршрут и полную стоимость перевозки.

а)

	A	B	C	D	E
A			3	1	
B			4		2
C	3	4			2
D	1				
E		2	2		

б)

	A	B	C	D	E
A			3	1	1
B			4		
C	3	4			2
D	1				
E	1		2		

в)

	A	B	C	D	E
A			3	1	4
B			4		2
C	3	4			2
D	1				
E	4	2	2		

г)

	A	B	C	D	E
A				1	
B			4		1
C		4		4	2
D	1		4		
E		1	2		

11. Постройте орграфы, соответствующие весовым матрицам.

а)

	A	B	C	D	E
A			3	1	
B	2		4		2
C	3				
D	1				
E			2		

б)

	A	B	C	D	E
A			5	1	1
B			6	4	
C	3	4			2
D		2			
E			3		

в)

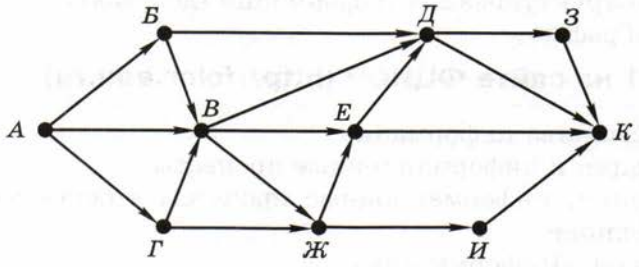
	A	B	C	D	E
A			3	1	4
B			4		2
C		4			2
D					
E	4		2		

г)

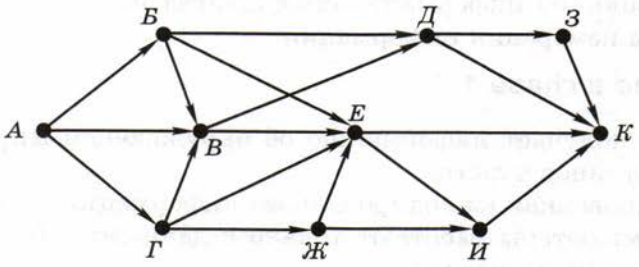
	A	B	C	D	E
A				1	
B			4		1
C	3	4		4	2
D	1	2	4		
E	1	1	2		

12. Для каждого из орграфов найдите количество различных маршрутов из вершины *A* во все остальные вершины.

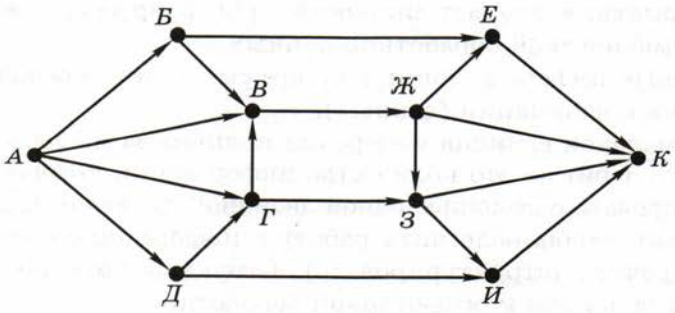
а)



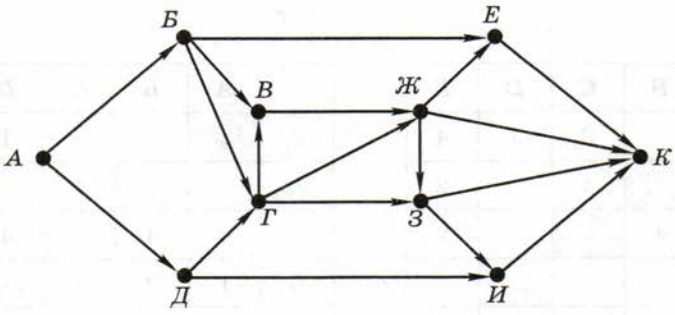
б)



в)



г)



Практические работы к главе 1

Работа № 1 «Оформление документа»
Работа № 2 «Структуризация информации (таблица, списки)»
Работа № 3 «Структуризация информации (деревья)»
Работа № 4 «Графы»

ЭОР к главе 1 на сайте ФЦИОР (http//fcior.edu.ru)

- Виды и свойства информации
- Информация и информационные процессы
- Информация, информационные процессы в обществе, природе и технике
- Что изучает «Информатика»
- Классификация информационных процессов
- Единицы измерения информации

Самое важное в главе 1

- Человек получает информацию об окружающем мире с помощью органов чувств.
- Зафиксированная (закодированная) информация — это данные. Компьютеры работают только с данными. Данные сохраняются на носителях.
- Информатика изучает широкий круг вопросов, связанных с автоматической обработкой данных.
- Основные информационные процессы — это передача и обработка информации (данных).
- Минимальная единица измерения количества информации — это бит. 1 бит — это количество информации, которое можно закодировать с помощью одной двоичной цифры (0 или 1).
- Для того чтобы облегчить работу с информацией, её нужно упорядочить (структурировать). Структурирование данных обязательно для компьютерной обработки.

Глава 2
Кодирование информации

§ 5
Язык и алфавит

Как записать информацию?

Для того чтобы хранить и передавать информацию, её необходимо как-то зафиксировать, например записать с помощью символов (знаков) на каком-то языке.

Язык — это система знаков, используемая для хранения, передачи и обработки информации.

Естественные языки (русский, английский и др.) сформировались в результате развития человеческого общества и используются для общения людей.

Сначала древние люди овладели устной речью. Поскольку человек может издавать и различать на слух не так много звуков, он стал комбинировать их, составляя слова, каждому из которых приписывался некоторый смысл.

Затем появилась необходимость записывать информацию, например, для передачи потомкам. В первое время жизненный опыт пытались зафиксировать в виде рисунков животных и предметов, затем пиктограмм (схематических изображений), иероглифов (рис. 2.1).

В большинстве современных языков используется **алфавитное письмо**, где каждый знак (или сочетание знаков) обозначает некоторый звук, так что с помощью небольшого набора знаков (алфавита) можно записать любые слова устной речи.

Алфавит — это набор знаков, который используется в языке.

Чаще всего подразумевается, что символы в алфавите расположены в определённом порядке.

Египетское письмо	
	Рука
	Дом
	Кобра
	Лев
	Вода
	Рот
	Мужчина
	Женщина

Иероглифы (Китай)	
日	Солнце
月	Луна
雨	Дождь
山	Гора
马	Лошадь
鱼	Рыба
人	Человек
女	Женщина

Рис. 2.1

В алфавите русского языка 33 буквы, в английском алфавите — 26. К алфавиту языка, вообще говоря, нужно отнести пробел (пропуск между словами), цифры (знаки для записи чисел), знаки препинания, скобки.

Мощность алфавита — это количество знаков в алфавите.

Например, алфавит, состоящий из 33 русских букв, 10 цифр, пробела и 12 знаков препинания (точка, запятая, точка с запятой, вопросительный и восклицательный знаки, тире, двоеточие, мно-

готочие, кавычки, круглые скобки) имеет мощность 56 (а если различать прописные и строчные буквы, то 89).

Слово — это последовательность символов алфавита, которая используется как самостоятельная единица и имеет определённое значение.

Из слов составляются **предложения**, каждое из которых выражает определённую законченную мысль (сообщение, порцию информации). В языке определяются правила построения слов (**грамматика**), правила построения предложений (**синтаксис**) и правила расстановки знаков препинания (**пунктуация**).

С точки зрения теории информации, сообщение — это любой набор знаков некоторого алфавита. Определим, сколько различных сообщений можно построить с помощью заданного количества знаков. Пусть, например, алфавит состоит из четырёх знаков: @ # $ %. С его помощью можно записать 4 разных сообщения из одного символа: @, #, $ и %. Теперь рассмотрим сообщения из двух знаков. Первый знак можно выбрать четырьмя способами, и для каждого из них есть 4 варианта выбора второго знака. Поэтому сообщений, состоящих из двух знаков, будет $4^2 = 16$:

@@	#@	$@	%@
@#	##	$#	%#
@$	#$	$$	%$
@%	#%	$%	%%

Рассуждая аналогично, получим, что трёхсимвольных сообщений будет $4^3 = 64$, а четырёхсимвольных — $4^4 = 256$ и т. д.

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

$$Q = M^N$$

Естественные и формальные языки

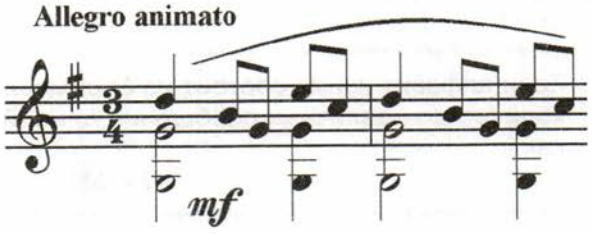
Вы знаете, что в естественных языках кроме правил есть и исключения. Кроме того, одно и то же слово может иметь различный смысл в зависимости от *контекста*, т. е. фрагмента текста, в котором оно употребляется. Например, в книге по географии слово «рукав», скорее всего, будет означать ответвление русла реки, а не деталь одежды. Существует некоторая «свобода понимания»: слова и выражения могут пониматься немного по-разному в зависимости от множества условий: опыта человека, его культуры, уровня образованности, настроения и т. п. Так кулинарные рецепты часто содержат совет «добавить соль по вкусу», который каждый выполняет по-своему.

В то же время, например, в научных публикациях такая ситуация недопустима, потому что смысл текста должен быть понят однозначно. В таких случаях используют языки специального типа, в которых каждое слово и словосочетание имеет чётко определенное единственное значение, и нет никаких исключений.

Формальный язык — это язык, в котором однозначно определяется значение каждого слова, а также правила построения предложений и придания им смысла.

Вот некоторые примеры формальных языков:

- математические формулы: $y = 3 \sin x + 1$;
- химические формулы и правила записи реакций: $2H_2 + O_2 = 2H_2O$;
- системы счисления (правила записи чисел с помощью специальных знаков — цифр): 12345, XXI;
- нотная запись:



- язык записи шахматных партий: 1. e2–e4 e7–e5...;

- язык программирования Паскаль:

```
program qq;
begin
  write('Привет, Вася!')
end.
```

Все формальные языки — **искусственные**. В отличие от естественных языков, которые формировались в течение многих веков и неотделимы от истории каждого народа и его культуры, формальные языки разрабатываются людьми для обмена информацией в специальных областях знаний. Например, нотная запись позволяет сохранить и передать музыкальное произведение. К формальным языкам часто относят эсперанто — искусственный язык, разработанный в конце XIX века польским врачом Л. М. Заменгофом для международного общения.

В таблице 2.1 проведено сравнение естественных и формальных языков.

Таблица 2.1

Естественные языки	Формальные языки
Сформировались в результате развития общества	Созданы людьми целенаправленно
Используются для общения в быту	Используются для обмена информацией в специальных областях знаний
Часто встречаются слова с неточным и неясным содержанием	Не допускаются слова с неточным и неясным содержанием
Значения отдельных слов и предложений зависят не только от них самих, но и от их окружения (контекста)	Значения отдельных слов и предложений не зависят от контекста
Встречаются синонимы (разные слова имеют одинаковый смысл)	Как правило, синонимов нет
Встречаются омонимы (одно слово может иметь несколько значений)	Омонимов нет
Нет строгих правил образования предложений	Правила образования предложений строго определены
Для многих правил существуют исключения	Нет исключений из правил

Вопросы и задания

1. Что такое язык?
2. Зачем нужны языки?
3. Какие языки называются естественными?
4. Что такое алфавит языка?
5. Как вы думаете, почему алфавиты большинства современных языков содержат небольшое число знаков?
6. Чем отличается алфавитное письмо от использования иероглифов?
7. Какие правила существуют в языке? Как они называются?
8. В каких областях требуется использование формальных языков?
9. Чем отличается формальный язык от естественного?
10. Что такое контекст? Почему меню, которое появляется при щелчке правой кнопкой мыши на объекте, называют контекстным?
11. Приведите примеры формальных языков, о которых не упоминалось в тексте учебника.
12. Объясните, почему любой язык программирования — это формальный язык.
13. Как вы думаете, почему люди не отказываются от естественных языков и не переходят на формальные во всех областях?
14. Как вы думаете, почему любой формальный язык не является универсальным и хорошо подходит для записи информации только в определённой области?
15. Выберите из табл. 2.1 те свойства естественных языков, которые затрудняют и не позволяют полностью автоматизировать перевод с одного языка на другой. Приведите примеры.

Подготовьте сообщение:

а) «Что такое алфавит?»
б) «Зачем нужны формальные языки?»
в) «Язык эсперанто»

§ 6 Кодирование

Как показано в главе 1 и в § 5, для хранения и передачи информации нужно записать её, зафиксировать на некотором языке (с помощью какого-то алфавита), т. е. закодировать. Это особенно важно в наше время, когда данные в компьютерных системах передаются, хранятся и обрабатываются в закодированном виде.

Кодирование — это представление информации в форме, удобной для её хранения, передачи и обработки. Правило такого преобразования называется **кодом**. Кодом называют также набор знаков закодированного сообщения.

В зависимости от конкретной задачи информация может кодироваться разными способами. Например, фраза «Привет, Вася!» может быть закодирована *транслитом* (так сокращённо называют транслитерацию — русский текст, записанный латинскими буквами): «Privet, Vasya!». Такой метод используют в электронных письмах, когда у одного собеседника (или у обоих) на компьютере нет поддержки русского языка. То же самое сообщение можно просто перевести на английский (или какой-то другой) язык, если собеседник не знает русского языка. А можно даже зашифровать: «Рсйгжу-!Гбта»». Шифрование — это один из способов кодирования, при котором нужно скрыть смысл сообщения от посторонних[1].

Для кодирования числовой информации в разных ситуациях тоже используют разные способы. Например, число 21 можно записать как XXI (в римской системе счисления) или «двадцать один» (в финансовых документах).



Самюэль Морзе (1791–1872)

Код Морзе

Долгое время для передачи сообщений по телеграфу и радио применялся код Морзе[2] (азбука Морзе), предложенный американским художником и изобретателем Самюэлем Морзе. В этом коде все буквы и цифры кодируются в виде различных последовательностей точек и тире (рис. 2.2).

---

[1] Это сообщение зашифровано с помощью шифра Цезаря. Попробуйте разгадать этот шифр и сформулировать правила кодирования и декодирования.

[2] Код Морзе применялся в британском флоте с 1865 г. для передачи сообщений с помощью флажков (днем) и фонарей (ночью). Для этой же цели использовали прожектора, у которых закрывали и открывали специальные жалюзи, а также сирены (для звуковой связи). С начала XX века код Морзе начали применять в радиосвязи.

Код Морзе для русских букв и цифр

А	•—	О	–––	Э	••–••
Б	–•••	П	•––•	Ю	••––
В	•––	Р	•–•	Я	•–•–
Г	––•	С	•••		
Д	–••	Т	–	1	•————
Е, Ё	•	У	••–	2	••–––
Ж	•••–	Ф	••–•	3	•••––
З	––••	Х	••••	4	••••–
Й	••	Ц	–•–•	5	•••••
И	•–––	Ч	–––•	6	–••••
К	–•–	Ш	————	7	––•••
Л	•–••	Щ	––•–	8	–––••
М	––	Ь, Ъ	–••–	9	––––•
Н	–•	Ы	–•––	0	–––––

Рис. 2.2

Код Морзе — *неравномерный*, т. е. коды символов могут быть разной длины. Для сокращения общего времени передачи буквы, которые встречаются чаще, имеют более короткие коды. Чтобы узнать, как часто встречается каждая буква в текстах, Морзе посетил типографию и подсчитал количество используемых литер с изображениями разных букв. Поэтому английская буква «E», которая встречается в текстах чаще всего, получила код •. Коды Морзе для русских букв совпадают с кодами похожих по звучанию английских букв, например коды букв «Л» и «L» одинаковы[1].

Чтобы отделить последовательности (коды букв) друг от друга, вводят еще один символ — пробел (пауза). Например, имя «Вася», закодированное с помощью кода Морзе, выглядит так:

•–– •– ••• •–•

Если бы не было разбивки на буквы, текст перестал бы расшифровываться однозначно. Например, сообщение •––•– можно было бы прочитать как ВА, АК, ПТ или даже ЕМЕТ.

[1] Поэтому код Морзе для русских букв менее эффективен.

Двоичное кодирование

Для передачи информации обязательно нужно, чтобы свойства носителя как-то изменялись. Самый простой используемый код должен содержать, по крайней мере, два разных знака. Такое кодирование называют **двоичным** (от слова «два»), оно используется практически во всех современных компьютерах.

Двоичное кодирование — это кодирование с помощью двух знаков.

Например, сообщение АБАВГБ может быть закодировано с помощью кодовой таблицы

А	Б	В	Г
00	01	10	11

следующим образом: 000100101101.

Кодирование чисел с помощью нулей и единиц впервые применил в своей (механической) вычислительной машине немецкий мыслитель Готфрид Вильгельм Лейбниц в конце XVII века. Затем, уже в середине XX века, двоичное кодирование информации стало повсеместно применяться для электронных компьютеров.

Чаще всего используется равномерный код, когда все символы исходного сообщения кодируются с помощью одинакового количества двоичных знаков. Каждый знак соответствует выбору одного из двух вариантов (0 или 1), поэтому несёт 1 бит информации.

Длина кода определяется количеством вариантов, которые нужно закодировать. Поскольку алфавит двоичного кода содержит 2 символа, применяя общую формулу (см. § 3), получаем количество различных сообщений длиной N битов:

$$Q = 2^N.$$



Готфрид Вильгельм Лейбниц (1646–1716)

Если заданное количество вариантов не равно степени числа 2, выбирают длину кода с запасом. Например, для кодирования номера спортсмена в интервале от 1 до 200 нужно использовать не меньше, чем 8 битов, поскольку

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

Если нужно передать информацию о номерах первых 20 спортсменов, пришедших к финишу, информационный объём такого сообщения будет равен $8 \cdot 20 = 160$ битов = 20 байтов.

Вопросы и задания

1. Что такое кодирование?
2. Зачем кодируют информацию?
3. Что такое код?
4. Какой алфавит используется в коде Морзе?
5. Какие буквы в коде Морзе имеют самые короткие коды? Почему?
6. Предложите, как можно изменить азбуку Морзе, чтобы сообщения на русском языке стали более короткими.
7. Запишите своё имя с помощью кода Морзе.
8. Почему в коде Морзе необходим символ-разделитель (пауза)?
9. В каком случае применяется транслитерация?
10. Где сейчас используются числа, записанные в римской системе счисления?
11. Как вы думаете, зачем в финансовых документах денежные суммы пишут прописью?
12. Какое кодирование называют двоичным?
13. Можно ли при двоичном кодировании использовать не 0 и 1, а другие знаки (например, буквы А и Б)?
14. Объясните, как при двоичном кодировании связаны длина сообщения и количество информации в нём.

Подготовьте сообщение

а) «Код Морзе»
б) «История двоичного кодирования»
в) «Код Грея»
г) «Шрифт Брайля»

Задачи

1. Сколько существует в коде Морзе различных последовательностей из точек и тире, длина которых равна 4 символа? 6 символов?
2. Сколько различных пятизначных чисел можно записать с помощью цифр 4 и 2?
3. В алфавите языка племени «тамба-амба» две буквы: Й и Ы. Сколько различных 11-буквенных слов можно образовать в этом языке?
4. Алфавит языка «амба-карамба» состоит из 5 букв. Сколько различных четырехбуквенных слов можно образовать в этом языке?
5. В языке племени «тумба-юмба» разрешены только четырёхбуквенные слова, которые можно образовывать из букв алфавита в любых комбинациях. Известно, что словарный запас языка составляет 81 слово. Какова мощность алфавита?
6. Некоторый язык содержит только трёхбуквенные слова, которые можно образовывать из букв его алфавита в любых комбинациях. Известно, что словарный запас языка составляет 216 слов. Какова мощность алфавита?
7. Какое наименьшее число символов должно быть в алфавите, чтобы с помощью всевозможных трёхбуквенных слов, состоящих из символов данного алфавита, можно было передать не менее 9 различных сообщений?
8. Световое табло состоит из лампочек. Каждая лампочка может находиться в одном из трех состояний («включено», «выключено» или «мигает»). Какое наименьшее количество лампочек должно находиться на табло, чтобы с его помощью можно было передать 18 различных сообщений?
9. Некоторое сигнальное устройство за одну секунду передаёт один из трёх сигналов. Сколько различных сообщений длиной в четыре секунды можно передать с помощью этого устройства?
10. Световое табло состоит из светящихся элементов, каждый из которых может гореть одним из двух различных цветов (или не гореть вообще). Сколько различных сообщений можно передать с помощью табло, состоящего из пяти таких элементов?
11. Для передачи сигналов на флоте используются специальные сигнальные флаги, вывешиваемые в одну линию (последовательность важна). Какое количество различных сообщений может передать корабль с помощью пяти сигнальных флагов, если на корабле имеются флаги четырёх различных видов (флагов каждого вида неограниченное количество)?
12. Вася и Петя передают друг другу сообщения, используя синий, красный и зелёный фонарики. Это они делают, включая по одному фонарику на одинаковое короткое время в некоторой последова-

тельности. Количество вспышек в одном сообщении — 3 или 4, между сообщениями — паузы. Сколько различных сообщений могут передавать мальчики?

13. Для кодирования 300 различных сообщений используются 5 последовательных цветовых вспышек. Вспышки одинаковой длительности, для каждой вспышки используется одна лампочка определённого цвета. Лампочки скольких цветов должны использоваться при передаче (укажите минимально возможное количество)?

*14. Некоторый алфавит содержит 4 различных символа. Сколько трёхбуквенных слов можно составить из символов этого алфавита, если символы в слове не могут повторяться?

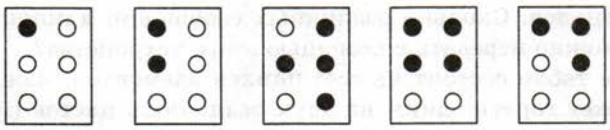
*15. В текстовом процессоре есть 5 кнопок, с помощью которых можно включать и выключать следующие режимы: жирный шрифт, курсив, подчёркивание, верхний индекс, нижний индекс. Сколько различных стилей оформления текста можно использовать?

16. Используя кодовую таблицу

А	Б	В	Г
00	01	10	11

закодируйте сообщение ГАВВАБ.

17. Шрифт Брайля — это специальный шрифт, с помощью которого незрячие люди могут читать. Для кодирования используются 6 точек, расположенных в два столбца. В каждой из них может быть выпуклость, которую человек воспринимает на ощупь. Коды Брайля первых букв русского алфавита (чёрная точка обозначает выпуклость):



Сколько различных символов можно закодировать с помощью кода Брайля?

18. Предложите какой-нибудь способ перехода от шрифта Брайля к двоичному кодированию.

19. В чём преимущества использования двоичного кодирования информации в современных компьютерах?

20. Сколько существует различных последовательностей из символов «плюс» и «минус» длиной ровно в пять символов?

21. На хранение целого числа отвели 12 битов. Сколько различных чисел можно закодировать таким образом?

22. Разведчик кодирует секретные сообщения, расставляя крестики и нолики в ячейки таблицы. Всего он может закодировать 512 сообщений. Сколько ячеек в таблице у разведчика?

23. Шахматная доска состоит из 8 столбцов и 8 строк. Какое минимальное количество битов потребуется для кодирования координат одной шахматной фигуры?

24. Какое минимальное количество битов потребуется для кодирования одного из натуральных чисел, меньших 60?

25. Для кодирования значений температуры воздуха (целое число в интервале от −50 до 40) используется двоичный код. Какова минимальная длина двоичного кода?

26. В сельскохозяйственном институте изучают всхожесть семян растений. Результатом одного измерения является целое число от 0 до 100%, которое записывается с помощью минимально возможного количества битов. Всего исследовано 60 партий семян. Определите информационный объём результатов наблюдений.

27. Обычный дорожный светофор без дополнительных секций подаёт шесть видов сигналов (непрерывные красный, жёлтый и зелёный, мигающие жёлтый и зелёный, мигающие красный и жёлтый одновременно). Электронное устройство управления светофором последовательно воспроизводит записанные сигналы. Подряд записано 100 сигналов светофора. Определите информационный объём этого сообщения.

28. В некоторой стране автомобильный номер длиной 6 символов составляется из заглавных букв (всего используется 12 букв) и десятичных цифр в любом порядке. Каждый символ кодируется одинаковым и минимально возможным количеством битов, а каждый номер — одинаковым и минимально возможным количеством байтов. Определите объём памяти, необходимый для хранения 32 автомобильных номеров.

29. В базе данных хранятся записи, содержащие информацию о датах. Каждая запись содержит три поля: год (число от 1 до 2100), номер месяца (число от 1 до 12) и номер дня в месяце (число от 1 до 31). Каждое поле записывается отдельно от других полей с помощью минимально возможного числа битов. Определите минимальное количество битов, необходимое для кодирования одной записи.

30. В соревнованиях по ориентированию участвуют 768 спортсменов. Специальное устройство регистрирует финиш каждого из участников, записывая его номер с использованием минимально возможного количества битов, одинакового для каждого спортсмена. Каков будет информационный объём сообщения (в байтах), записанного устройством, после того как финишируют 200 спортсменов?

Декодирование

Декодирование — это восстановление информационного сообщения из последовательности кодов.

Например, закодированное сообщение

•—• • — •—• ——— •——

можно восстановить, используя код Морзе «в обратную сторону»: в этой строке закодирована фамилия «Петров».

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

Пример 1. Пусть для кодирования первых 5 букв русского алфавита используется таблица:

А	Б	В	Г	Д
000	10	01	110	001

Это неравномерный код, поскольку в нём есть двух- и трёхсимвольные кодовые слова. Построим для этой кодовой таблицы дерево, в котором от каждого узла (кроме листьев) отходят два ребра, помеченные цифрами 0 и 1. Чтобы найти код символа, нужно пройти по стрелкам от корня дерева к нужному листу, выписывая метки стрелок, по которым мы переходим (рис. 2.3).

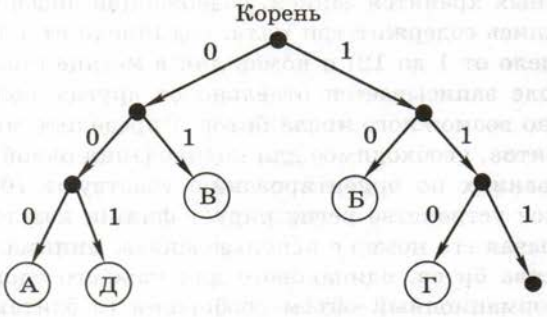


Рис. 2.3

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

Г	
110	0000100110

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

Г	А	
110	000	0100110

Аналогично декодируем всё сообщение:

Г	А	В	Д	Б
110	000	01	001	10

Пример. 2. Рассмотрим другую кодовую таблицу:

А	Б	В	Г	Д
000	01	10	011	100

Здесь условие Фано не выполняется, поскольку код буквы Б (01) является началом кода буквы Г (011), а код буквы Д (100) начинается с кода буквы В (10). Дерево для этой кодовой таблицы выглядит так (рис. 2.4).

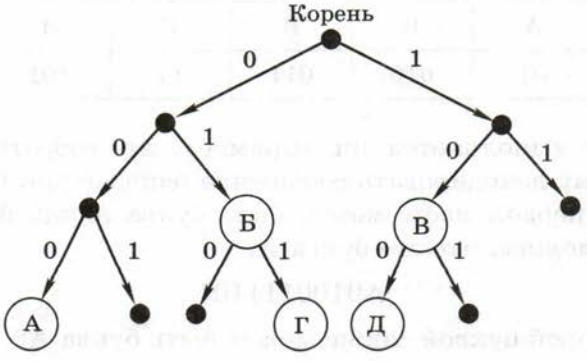


Рис. 2.4

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

	В
0110001101	10

Вторая буква с конца — Б (код 01):

	Б	В
01100011	01	10

И так далее:

Б	Д	Г	Б	В
01	100	011	01	10

В общем случае (если код не является ни префиксным, ни постфиксным) декодировать сообщение удаётся только перебором вариантов.

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

А	Б	В	Г	Д
01	010	011	11	101

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

А0100111101

Тогда второй буквой также может быть буква А:

АА00111101.

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

АБ0111101.

Третьей буквой может быть А:

АБА11101,

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

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

АБВ1101,

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

Кроме того, на первом месте может стоять буква Б:

Б100111101,

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

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

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

1. Определим все последовательности, которые совпадают с началом какого-то кодового слова и одновременно с концом какого-то кодового слова; в данном случае это две последовательности:
   0 (начало кода буквы А и конец кода буквы Б)
   и 1 (начало кода буквы Г и конец кода буквы Д).
   Последовательности 01 и 11 не учитываем, потому что они совпадают с кодами букв А и Г.

2. Добавим к этому множеству {0, 1} пустую строку, которую обычно обозначают буквой Λ (прописная греческая буква «лямбда»); элементы полученного множества {Λ, 0, 1} становятся узлами графа (рис. 2.5).

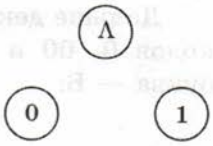


Рис. 2.5

3. Соединим узлы дугами (направленными рёбрами) по такому правилу: два узла X и Y соединяются дугой, если последовательная запись кода узла X, кода некоторой буквы (или нескольких букв) и кода узла Y даёт код ещё одной буквы (рис. 2.6).

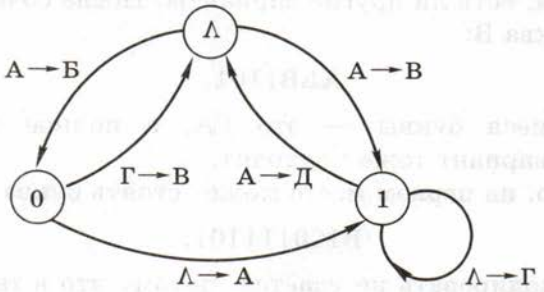


Рис. 2.6

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

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

В нашем графе есть несколько таких циклов, например:

- цикл Λ0Λ, соответствующий сообщению ΛА0ГΛ = 01011; это сообщение может быть расшифровано как АВ и как БГ;
- цикл Λ1Λ, соответствующий сообщению ΛА1АΛ = 01101; это сообщение может быть расшифровано как АД и как ВА;

- цикл Λ01Λ, соответствующий сообщению ΛA01AΛ = 010101; это сообщение может быть расшифровано как AAA и как БД.

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

Пример 4. Существуют коды, для которых условия Фано не выполняются, но все сообщения однозначно декодируются. В кодовой таблице

А	Б	В
0	11	010

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

Проверим, можно ли однозначно декодировать сообщения, построенные с помощью такого кода. Множество последовательностей, которые совпадают с началом и концом кодовых слов, состоит из пустой строки и единицы: {Λ, 1}. Граф, построенный с помощью приведённого выше алгоритма, содержит два узла и одну петлю (рис. 2.7).

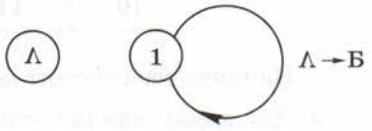


Рис. 2.7

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

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

Иногда при кодировании и декодировании происходит искажение сообщения. Например, известно, что перевод художественных текстов (особенно стихов) на другой язык и затем обратный перевод могут изменить их до неузнаваемости.

Вопросы и задания

1. Что такое декодирование?
2. Всегда ли удаётся однозначно декодировать сообщение? В каких случаях это может быть не так?
3. Перечислите достаточные условия, при которых можно однозначно декодировать сообщение с неравномерным кодом.
4. В каких случаях для декодирования приходится использовать перебор вариантов?

Задачи

1. Расшифруйте сообщение, записанное с помощью кода Морзе, которое используется как международный сигнал бедствия:
   ••• — — — •••.
2. Покажите с помощью дерева, что кодовая таблица из примера 2 удовлетворяет «обратному» условию Фано.
3. Для кодирования сообщения используется таблица

А	Б	В	Г	Д
10	11	001	010	011

   Найдите все способы декодирования сообщения 1111001011.
4. Для кодирования сообщения используется таблица

А	Б	В	Г	Д
01	11	100	010	110

   Найдите все способы декодирования сообщения 1111001001100.
5. Для кодирования сообщения используется таблица

А	Б	В	Г	Д
0	11	101	110	111

   Найдите все способы декодирования сообщения 1111001010.

6. Для кодирования сообщения используется таблица

А	Б	В	Г	Д
0	10	1	110	111

Найдите все способы декодирования сообщения 01110011.

7. Для кодирования сообщения используется таблица

A	B	C	D	E
000	01	100	10	011

Декодируйте сообщение 0110100011000.

8. Для кодирования сообщения, состоящего только из букв A, B, C, D и E, используется неравномерный двоичный код:

A	B	C	D	E
000	11	01	001	10

Какие из сообщений были переданы без ошибок:

1) 110000010011110
2) 110000011011110
3) 110001001001110
4) 110000001011110

***9.** Для передачи по каналу связи сообщения, состоящего только из букв А, Б, В, Г, решили использовать неравномерный код: А = 0, Б = 10, В = 110. Как нужно закодировать букву Г, чтобы длина кода была минимальной и допускалось однозначное разбиение кодированного сообщения на буквы?

***10.** Для передачи по каналу связи сообщения, состоящего только из букв А, Б, В, Г, решили использовать неравномерный код: А = 0, Б = 100, В = 101. Как нужно закодировать букву Г, чтобы длина кода была минимальной и допускалось однозначное разбиение кодированного сообщения на буквы?

***11.** Для передачи по каналу связи сообщения, состоящего только из букв А, Б, В, Г, решили использовать неравномерный код: А = 01, Б = 1, В = 001. Как нужно закодировать букву Г, чтобы длина кода была минимальной и допускалось однозначное разбиение кодированного сообщения на буквы?

*12. Для передачи по каналу связи сообщения, состоящего только из букв А, Б, В, Г, решили использовать неравномерный код: А = 0, Б = 100, В =110. Как нужно закодировать букву Г, чтобы длина кода была минимальной и допускалось однозначное разбиение кодированного сообщения на буквы?

*13. Для передачи по каналу связи сообщения, состоящего только из букв А, Б, В, Г, решили использовать неравномерный код: А = 00, Б = 11, В = 100 и Г = 10. Определите, допускает ли такой код однозначное декодирование сообщения. Выполняется ли для него условие Фано?

§ 7 Дискретность

Аналоговые и дискретные сигналы

Как вы уже знаете, информация передаётся в закодированном виде с помощью сигналов. Согласно определению из § 2, сигнал — это изменение свойств носителя, которое используется для передачи информации. Изменение выбранного свойства (например, силы тока, напряжения, освещённости) во времени можно описать в виде функции. Далее такую функцию тоже будем называть сигналом, как это принято в электронике и вычислительной технике.

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

Элементы электронных устройств, как правило, обмениваются данными с помощью электрических сигналов; для получения информации приёмник должен измерить этот сигнал (чаще всего — напряжение на контактах). В таких устройствах, как радиоприёмник и микрофон, изменение электрического сигнала может произойти в любой момент и быть любым (в пределах допустимого диапазона). Такие сигналы называют **аналоговыми**.

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

Органы чувств человека воспринимают информацию в аналоговой форме: свет, звуковые волны, вкус, запах и т. п. Поэтому

раньше большинство технических устройств для работы с информацией (телефоны, магнитофоны, фотоаппараты) тоже было аналоговым.

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

Дело в том, что при передаче сигнала всегда есть помехи, которые искажают его значения. В большинстве случаев эти искажения — случайные ошибки, не поддающиеся учёту. Фактически приёмник получает не исходный сигнал, посланный источником (сплошная линия на рис. 2.8), а искажённый (штриховая линия).

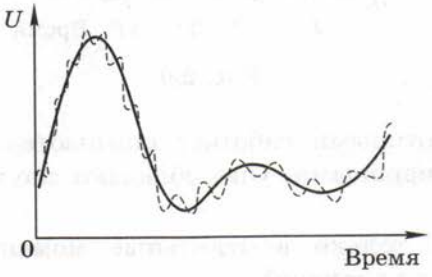


Рис. 2.8

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

Если использовать аналоговые компьютеры, мы будем при каждом расчёте с одинаковыми исходными данными получать несколько отличающиеся результаты. Кроме того, при копировании аналоговая информация искажается (например, при каждом копировании звукозаписи на магнитной ленте качество копии ухудшается).

Эта ситуация не устраивала инженеров, разрабатывающих компьютеры, и они нашли интересное решение: если не удаётся точно измерить сигнал, нужно вообще отказаться от его измерения, а просто через некоторый интервал времени определять,

в каком из двух состояний находится сигнал (эти состояния можно обозначить как 1 и 0)[1]. При использовании такого подхода мы получаем огромное преимущество: при небольших помехах искажение сигнала не влияет на передачу данных: если напряжение выше некоторого порога U_1, считается, что сигнал равен 1, а все сигналы, меньшие другого порога U_0, считаются равными нулю (рис. 2.9).

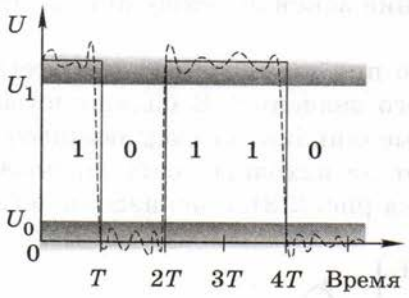


Рис. 2.9

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

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

Дискретный (цифровой) сигнал — это последовательность значений, каждое из которых принадлежит некоторому конечному множеству.

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

Так как каждому значению дискретного сигнала всегда можно поставить в соответствие определённый знак, такой сигнал

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

можно рассматривать как сообщение, записанное с помощью конечного набора знаков (алфавита).

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

Дискретизация

Поскольку данные в компьютерах передаются с помощью дискретных сигналов, компьютеры могут хранить и обрабатывать только **дискретную информацию**, т. е. такую, которая может быть записана с помощью конечного количества знаков некоторого алфавита. Поэтому для ввода любых данных в компьютер их нужно перевести в дискретный код.

Дискретность означает, что мы представляем нечто целое (непрерывное) в виде набора отдельных элементов. Например, картина художника — это аналоговая (непрерывная) информация, а мозаика, сделанная на её основе (рисунок из кусочков разноцветного стекла), — дискретная. Множество вещественных чисел непрерывно (между любыми двумя различными числами есть ещё бесконечно много других), а множество целых чисел дискретно.

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

Всем известное иррациональное число π содержит бесконечное количество знаков в дробной части. Если мы хотим записать, чему равно π, необходимо остановиться на каком-то знаке, отбросив остальные, например: $\pi \approx 3{,}14$. Таким образом, мы перешли к дискретной информации, потому что рассматриваем только числа с шагом 0,01 — точки на числовой оси (рис. 2.10).

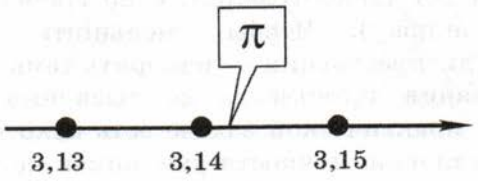
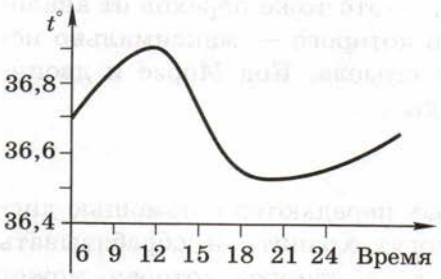


Рис. 2.10

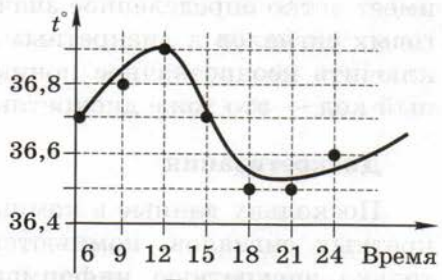
Изменение высоты столбика термометра — это аналоговая информация, а *записанная* температура, округлённая до десятых долей градуса (например, 36,6°), — дискретная (рис. 2.11).



Аналоговая информация

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

Дискретная информация



Дискретизация

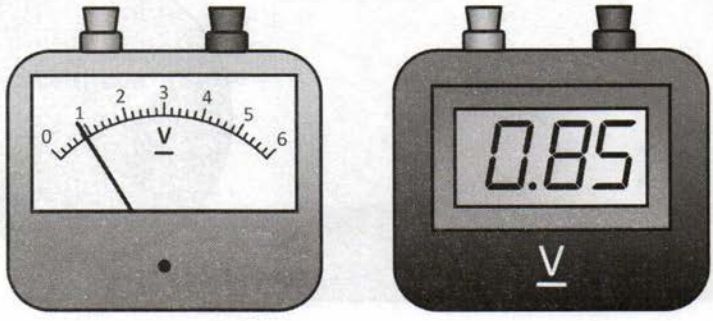
Рис. 2.11

Дискретность состоит в том, что записанные значения температуры изменяются скачкообразно (через 0,1°), — это *дискретизация по уровню*, или **квантование**. Кроме того, обычно температуру больного измеряют не непрерывно, а несколько раз в день — появляется *дискретизация по времени*.

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

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

Ещё один пример — аналоговые («стрелочные») и цифровые вольтметры, которые измеряют одну и ту же величину, но выводят результат измерения в разном виде (рис. 2.12).



Аналоговая информация Дискретная информация

Рис. 2.12

Теперь подумаем, как записать аналоговую величину, которая может принимать бесконечное множество значений. Вы уже знаете, что с помощью N знаков алфавита, состоящего из M символов, можно закодировать $Q = M^N$ разных сообщений. Поэтому теоретически для записи аналоговой величины придётся использовать бесконечное число знаков.

Итак, когда мы хотим записать (зафиксировать) информацию с помощью какого-то алфавита, нужно переходить к дискретному представлению. С одной стороны, это делает более надёжной передачу данных (если обе стороны одинаково понимают используемые знаки). С другой стороны, при дискретизации часть информации теряется.

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

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

Идеальная непрерывность существует только в теории. Мы считаем дерево, пластмассу, металл непрерывными, но на самом деле они состоят из отдельных молекул, расположенных на некотором расстоянии друг от друга, — это значит, что вещество дискретно. Иллюстрация в книге кажется нам сплошной, но при сильном увеличении видно, что она строится из отдельных точек (имеет «растр») (рис. 2.13).

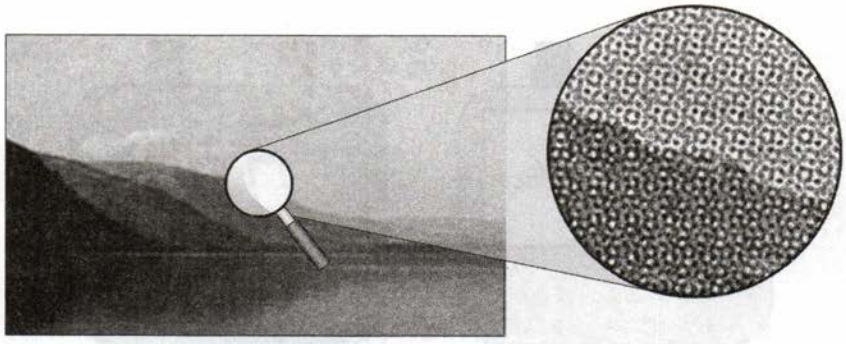


Рис. 2.13

«Плёночная» фотография считается аналоговой, но при увеличении снимка с фотоплёнки нельзя бесконечно получать все новые и новые детали — предел «уточнения» определяется величиной зерна светочувствительного материала.

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

Вопросы и задания

1. Что такое аналоговый сигнал?
2. Какие бытовые устройства работают с аналоговыми сигналами?
3. Какие системы связи используют аналоговые сигналы, а какие — дискретные?
4. Что такое аналоговые компьютеры? Почему они вышли из употребления?
5. Почему при использовании аналоговой техники передача информации всегда происходит с искажениями?

*6. Как вы думаете, почему аналоговый сигнал нельзя полностью «очистить» от помех? Какую дополнительную информации нужно иметь, чтобы эта задача могла быть частично решена?

7. Что такое дискретный сигнал?
8. Почему с помощью дискретного сигнала можно передавать информацию практически без искажений? Искажается ли такой сигнал под воздействием помех?
9. Сигнал изменяется в моменты времени, кратные 1 секунде, и может принимать одно из 16 возможных значений. Является ли такой сигнал дискретным?

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

11. Что такое дискретизация? Приведите примеры.
12. Что такое дискретизация по времени и дискретизация по уровню?
13. Приведите пример дискретизации сигнала только по уровню, только по времени. Нарисуйте графики таких процессов.
14. Объясните связь между дискретностью сигнала и алфавитным способом записи информации.
15. Почему при дискретизации, как правило, происходит потеря информации? В каких случаях потери информации не будет?
16. Как можно уменьшить потери информации при дискретизации?
17. Почему не стоит стремиться максимально уменьшить эти потери?
18. Приведите примеры, когда одна и та же информация может быть представлена в аналоговой и в дискретной форме.
19. Объясните фразу: «При увеличении точности дискретизации свойства непрерывной и дискретной информации практически совпадают».

Подготовьте сообщение

а) «Аналоговые вычислительные машины»
б) «Аналоговые и дискретные измерительные устройства»
в) «Непрерывность и дискретность в математике»
г) «Непрерывность и дискретность в природе»

§ 8
Алфавитный подход к измерению количества информации

Представьте себе, что вы много раз бросаете монету и записываете результат очередного броска как 1 (если монета упала гербом) или 0 (если она упала «решкой»). В результате получится некоторое *сообщение* — цепочка нулей и единиц, например 0101001101001110. Вы наверняка поняли, что здесь используется *двоичное* кодирование — это сообщение написано на языке, *алфавит* которого состоит из двух знаков (символов): 0 и 1. Как вы знаете из § 3, каждая двоичная цифра несёт 1 бит информации, поэтому полная информация в сообщении 0101001101001110 равна 16 битов.

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

Алфавитный подход к измерению количества информации состоит в следующем:

1) определяем мощность алфавита M (количество символов в алфавите);
2) по таблице степеней числа 2 определяем количество битов информации i, приходящихся на каждый символ сообщения, — *информационную ёмкость* (*объём*) *символа*:

M, символов	2	4	8	16	32	64	128	256	512	1024
i, битов	1	2	3	4	5	6	7	8	9	10

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

$$I = N \cdot i.$$

Обратим внимание на две важные особенности алфавитного подхода.

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

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

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

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

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

Для вычисления информационного объёма текста чаще всего применяют именно алфавитный подход. Например, пусть требуется оценить количество информации в 10 страницах текста (на каждой странице 32 строки по 64 символа) при использовании алфавита из 256 символов. Задача решается так:

1) определяем информационную ёмкость одного символа: так как $256 = 2^8$, один символ несёт $i = 8$ битов, или 1 байт информации;
2) считаем количество символов на одной странице, в данном случае удобно использовать степени числа 2 ($32 = 2^5$, $64 = 2^6$): $2^5 \cdot 2^6 = 2^{11}$ символов на странице;

3) находим общее количество символов на 10 страницах:
$N = 10 \cdot 2^{11}$ символов;

4) определяем информационный объём всего текста:
$I = N \cdot i = 10 \cdot 2^{11} \cdot 1$ байтов $= 10 \cdot 2^{11}$ байтов $=$
$= 10 \cdot 2^{11} \cdot (1/2^{10}$ Кбайт$) = 20$ Кбайт.

Вопросы и задания

1. В чём состоит алфавитный подход к измерению количества информации?
2. Приведите примеры ситуаций, когда смысл информации значения не имеет, а важен только её объём.
3. Учитывается ли при алфавитном подходе частота встречаемости символов в тексте?
4. Выберите все утверждения, справедливые для алфавитного подхода:
   а) количество информации зависит от длины сообщения;
   б) количество информации зависит от мощности алфавита;
   в) чем больше мощность алфавита, тем больше количество информации;
   г) важен смысл сообщения;
   д) сообщение должно быть понятно для приёмника;
   е) разные символы могут нести разное количество информации.
5. Технический документ перевели с одного языка на другой (считаем, что это было сделано максимально близко к тексту). Изменился ли смысл документа? Изменился ли его объём? Обоснуйте ответ.
6. Как вы думаете, почему компьютеру легко извлечь несколько предложений с конкретных страниц документа, но трудно составить аннотацию к документу?

Задачи

1. Сообщение состоит из 100 символов, используется алфавит, состоящий из 64 символов. Каков информационный объём этого сообщения?
2. Дан текст из 600 символов. Известно, что символы берутся из таблицы размером 16 × 32, в которой все ячейки заполнены разными символами. Определите информационный объём текста в битах.
3. Для записи текста использовался алфавит, состоящий из 32 символов. Каждая страница текста содержит 32 строки. Информационный объём сообщения, состоящего из 5 страниц, составил 6400 байтов. Сколько символов в каждой строке текста?

4. Страница текста содержит 30 строк по 60 символов в каждой. Сообщение, состоящее из 4 страниц текста, имеет информационный объём 6300 байтов. Какова мощность алфавита?
5. Мощность алфавита равна 256. Сколько Кбайт памяти потребуется для сохранения 160 страниц текста, содержащего в среднем 192 символа на каждой странице?
6. Мощность алфавита равна 64. Сколько Кбайт памяти потребуется, чтобы сохранить 128 страниц текста, содержащего в среднем 256 символов на каждой странице?
7. Секретарь может набирать текст со скоростью 256 символов в минуту. Сколько Кбайт информации он сможет ввести в компьютер за 10 минут, если используется алфавит из 256 символов?
8. Для кодирования секретного сообщения используются 12 специальных знаков. При этом символы кодируются одним и тем же минимально возможным количеством битов. Чему равен информационный объём сообщения длиной в 256 символов?
9. Для кодирования нотной записи используются 7 знаков-нот. Каждая нота кодируется одним и тем же минимально возможным количеством битов. Чему равен информационный объём (в битах) сообщения, состоящего из 180 нот?
10. Объём сообщения равен 7,5 Кбайт. Известно, что данное сообщение содержит 7680 символов. Какова мощность алфавита?
11. Объём сообщения равен 12 Кбайт. Сообщение содержит 16 384 символа. Какова мощность алфавита?
12. Объём сообщения, содержащего 4096 символов, равен 1/512 мегабайта. Какова мощность алфавита, с помощью которого записано это сообщение?
13. Два текста содержат одинаковое количество символов. Первый текст составлен в алфавите мощностью 16 символов, а второй текст — в алфавите из 256 символов. Во сколько раз количество информации во втором тексте больше, чем в первом?
14. Алфавит языка первого племени содержит 8 знаков, а алфавит языка второго племени — 16 символов. Племена обменялись сообщениями, состоящими из одинакового количества символов. Известно, что сообщение второго племени содержало 128 байтов информации. Каков информационный объём сообщения первого племени?

*15. Два текста содержат одинаковое количество символов, но информационный объём второго текста в 1,5 раза больше, чем информационный объём первого. Определите мощности алфавитов, если известно, что в обоих текстах число символов меньше 10, и на каждый символ приходится целое число битов.

***16.** Два текста имеют одинаковый информационный объём, но количество символов во втором тексте в 3,5 раза больше, чем в первом. Определите мощности алфавитов, если известно, что в обоих текстах число символов меньше 200, и на каждый символ приходится целое число битов.

§ 9
Системы счисления

Система счисления — это правила записи чисел с помощью специальных знаков — цифр, а также соответствующие правила выполнения операций с этими числами.

Первоначально люди считали на пальцах — это самый простой способ, который используется и сейчас. Один загнутый (или отогнутый) палец обозначал единицу (один день, одного человека, одного барана и т. п.). Такая система счисления называется **унарной** (от лат. *unus* — один). В качестве цифр унарной системы можно использовать камешки, узелки, счётные палочки (как в начальной школе), зарубки на дереве (как делал Робинзон Крузо) или на кости, чёрточки на бумаге, точки и другие одинаковые знаки или предметы.

С помощью унарной системы можно записывать только натуральные числа, причём запись больших чисел получается очень длинной (представьте себе, как записать миллион). Цифра любой позиции числа, записанного в унарной системе, всегда обозначает единицу, поэтому это одна из **непозиционных систем счисления**.

Непозиционная система счисления — это такая система счисления, в которой значение цифры не зависит от её места в записи числа.

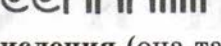
К непозиционным относится и **десятичная египетская система счисления**. Египтяне ввели 7 знаков-иероглифов, которые обо-

значали степени числа 10 (чёрточка, хомут, верёвка, лотос, палец, лягушка, человек) (рис. 2.14).



Рис. 2.14

В этой системе, например, число 235 записывалось как



В **римской системе счисления** (она также считается непозиционной) в качестве цифр используются латинские буквы: I обозначает 1, V — 5, X — 10, L — 50, C — 100, D — 500, M — 1000. Единицы, десятки, сотни и тысячи кодировались отдельными группами, например:

$$2368 = 2000 + 300 + 60 + 8 = (1000 + 1000) + (100 + 100 + 100) + (50 + 10) + (5 + 1 + 1 + 1) = \text{MMCCCLXVIII}.$$

Больше трёх одинаковых цифр подряд не ставили, поэтому число 4 записывали как IV. В такой записи меньшая цифра (I) стоит перед большей (V), поэтому она вычитается из неё. То есть:

$$\text{IV} = 5 - 1 = 4.$$

Аналогично записывались числа 9, 40, 90, 400 и 900:

$$\text{IX} = 10 - 1 = 9,\ \text{XL} = 50 - 10 = 40,\ \text{XC} = 100 - 10 = 90,$$
$$\text{CD} = 500 - 100 = 400,\ \text{CM} = 1000 - 100 = 900.$$

Из-за этой особенности римскую систему нельзя считать полностью непозиционной, потому что значение меньшей цифры, стоящей слева от большей, меняется на отрицательное.

У римской системы есть несколько серьёзных недостатков:

- можно записывать только натуральные числа (что делать с дробными и отрицательными?);
- чтобы записывать большие числа, необходимо вводить всё новые и новые цифры (иногда использовались цифры с подчёркиванием или чертой сверху, что обозначало увеличение в 1000 раз: $\underline{\text{V}}$ — 5000, $\underline{\text{X}}$ — 10 000 и т. д.);
- сложно выполнять арифметические действия.

Сейчас римская система применяется для нумерации веков (XXI век), месяцев, глав в книгах, на циферблатах часов (например, на Спасской башне Московского Кремля).

В **славянской системе счисления** в качестве цифр использовались буквы алфавита, над которыми ставился знак ҃ («титло») (рис. 2.15).



Рис. 2.15

Если в ряду стояло несколько цифр, знак «титло» ставился только у первой. Старшие цифры записывались *справа* от младших, например, число 11 записывалось как ѢI. Славянская система счисления используется на циферблате часов Суздальского кремля.

Вопросы и задания

1. Как можно закончить фразу: «Система счисления — это...»?
2. Что такое унарная система счисления? Приведите примеры.
3. Какие *недостатки* имеет унарная система счисления?
4. Что такое непозиционная система счисления?
5. Какие системы счисления относятся к непозиционным?
6. Какие цифры используются в римской системе? Что они означают?
7. Можно ли называть римскую систему полностью непозиционной? Обоснуйте ответ.
8. Какое наибольшее число можно записать в классической римской системе счисления?
9. Где сейчас используется римская система?
10. Перечислите недостатки римской системы счисления. Как вы думаете, почему её не используют в компьютерах?

Подготовьте сообщение

а) «Где применяется римская система счисления?»
б) «Славянская система счисления»
в) «Системы счисления разных народов»
г) «Система остаточных классов»

Задачи

1. Переведите в римскую систему числа: 12 345, 2999, 2444, 2888, 3777.
2. Переведите в десятичную систему числа: MCDXCIX, MMDCCXLVII, MDCXCIX.
3. Запишите в славянской системе числа: 15, 25, 38, 137, 596.

§ 10
Позиционные системы счисления

Основные понятия

Позиционная система счисления — это такая система счисления, в которой значение цифры («вес») полностью определяется её местом (позицией) в записи числа.

Пример позиционной системы счисления — привычная нам **десятичная система**. В числе 6375 цифра 6 обозначает тысячи (т. е. 6000), цифра 3 — сотни (300), цифра 7 — десятки (70), а цифра 5 — единицы:

$$6375 = 6 \cdot 1000 + 3 \cdot 100 + 7 \cdot 10 + 5 \cdot 1.$$

Алфавит системы счисления — это используемый в ней набор цифр.
Основание системы счисления — это количество цифр в алфавите (мощность алфавита).

В десятичной системе основание — 10, алфавит состоит из 10 цифр: 0, 1, 2, 3, 4, 5, 6, 7, 8 и 9. Число 10, вероятно, было выбрано потому, что люди сначала использовали для счета свои 10 пальцев на руках.

Разряд — это позиция цифры в записи числа. Разряды в записи целых чисел нумеруются с нуля справа налево.

В числе 6375 цифра 6 стоит в третьем разряде (тысячи, 10^3), 3 — во втором разряде (сотни, 10^2), 7 — в первом (десятки, 10^1), а 5 — в нулевом (единицы, 10^0). Не забывайте, что любое число (кроме нуля!) в нулевой степени равно 1. Поэтому

разряды → 3 2 1 0

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

Это так называемая **развёрнутая форма записи числа**. Из этой записи видно, что последняя цифра 5 — это остаток от деления числа на 10 (все остальные слагаемые делятся на 10); число, составленное из двух последних цифр (75), — это остаток от деления исходного числа на $100 = 10^2$ и т. д. Поэтому все числа, делящиеся на 100 без остатка, оканчиваются на два нуля.

Чтобы определить число, записанное в позиционной системе счисления, нужно значение каждой цифры умножить на основание системы счисления в степени, равной разряду этой цифры, и сложить полученные величины.

Число 6375 можно представить в другой форме — по *схеме Горнера*:

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

Эта форма позволяет найти число, используя только умножение и деление (без возведения в степень).

Кроме десятичной системы на практике используются ещё несколько позиционных систем:

- двоичная, восьмеричная и шестнадцатеричная в компьютерной технике;
- двенадцатеричная английская система мер (1 фут = 12 дюймов, 1 шиллинг = 12 пенсов);
- шестидесятеричная система измерения времени (1 час = = 60 минут, 1 минута = 60 секунд).

Целые числа

Теперь можно записать аналогичные выражения для системы счисления с любым натуральным основанием $p > 1$. Её алфавит состоит из p цифр[1] от 0 до $p - 1$, т. е. «старшая» (наибольшая) цифра в позиционной системе счисления на единицу меньше, чем основание.

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

разряды → 3 2 1 0

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

или с помощью схемы Горнера:

$$a_3a_2a_1a_0 = ((a_3 \cdot p + a_2) \cdot p + a_1) \cdot p + a_0.$$

Оба способа можно использовать для **перевода числа из любой позиционной системы в десятичную систему**. Например, пусть число 1234_5 записано в пятеричной системе счисления (с основанием 5). Нижний индекс 5 в записи 1234_5 обозначает основание системы счисления (для десятичной системы основание не указывают). Тогда:

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

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

Схема Горнера очень удобна для обработки данных при вводе чисел с клавиатуры, когда цифры числа вводятся последовательно, начиная с первой, и их количество заранее неизвестно.

Развёрнутую запись числа можно использовать для обратного перехода, от **десятичной системы к системе с основанием p**. Действительно, из формулы

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

следует, что a_0 — это остаток от деления исходного числа на основание p. Если мы разделим исходное число на p и отбросим остаток, мы получим:

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

Теперь легко найти a_1 — это последняя цифра получившегося числа, которая, как мы знаем, равна остатку от его деления на p.

[1] При $p > 10$ используются также и латинские буквы, но об этом далее.

Разделив новое получившееся число на p и отбросив остаток, получим число

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

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

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

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

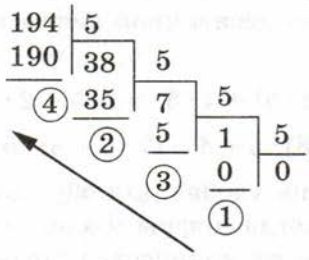
Таким образом, мы нашли последнюю цифру — 4. Частное равно 38, повторяем ту же операцию:

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

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

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

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



Ответ: 1234_5.

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

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

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

Пример 1. Зная десятичное число и его запись в некоторой позиционной системе счисления, можно найти основание этой системы. Пусть, например, число 71 в некоторой системе с основанием x записывается как 56_x. Представим это число в развёрнутой форме:

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

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

Пример 2. В более сложных случаях может получиться алгебраическое уравнение второй (или ещё более высокой) степени. Например, пусть то же число 71 в некоторой системе с основанием x записывается как 155_x. Представим это число в развёрнутой форме:

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

Решая уравнение $71 = x^2 + 5\,x + 5$ относительно неизвестного x, получаем два решения: $x_1 = -11$ и $x_2 = 6$. Искомое основание положительно, поэтому правильный ответ — 6.

Пример 3. Если запись числа в системе счисления задана не полностью, решений может быть несколько. Например, найдём все основания систем счисления, в которых запись десятичного числа 24 оканчивается на 3. Здесь удобно использовать схему Горнера, из которой сразу следует

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

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

Пример 4. Найдём все десятичные числа, не превосходящие 40, запись которых в системе счисления с основанием 4 оканчивается на 11. Используя схему Горнера, находим, что все интересующие нас числа имеют вид

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

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

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

1. ААААА
2. ААААО
3. ААААУ
4. АААОА
...

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

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

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

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

$$139 = 12011_3.$$

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

Вопросы и задания

1. Какие системы счисления называют позиционными?
2. Каким термином называется количество цифр в алфавите позиционной системе счисления?
3. Что такое разряд? Как нумеруются разряды?
4. Как связан в позиционной системе «вес» цифры и разряд, в котором она стоит?
5. Как вы думаете, почему в быту мы чаще всего используем десятичную систему?
6. Какие позиционные системы счисления используются сейчас на практике?
7. Чем хороша схема Горнера с точки зрения вычислений?
8. Как перевести число из любой позиционной системы в десятичную?
9. Какие цифры входят в алфавит девятеричной системы?
10. Как вы думаете, можно ли использовать систему счисления с основанием 1 000 000? В чём могут быть проблемы?
11. Сформулируйте алгоритм перевода числа из семеричной системы в десятичную.
12. Сформулируйте алгоритм перевода числа из десятичной системы в семеричную.
13. Как по записи числа в пятеричной системе сразу увидеть, делится ли оно на 5? на 25? на 125?

Задачи

1. Запишите число 12 345 в развёрнутой форме и в виде схемы Горнера.
2. Какое минимальное основание должно быть у системы счисления, чтобы в ней существовали числа 123, 463, 153 и 455? Ответ обоснуйте.
3. Выберите наибольшее число: 11_2, 11_7, 11, 11_{12}, 11_{16}, 11_{25}.
4. Переведите числа 345_6, 345_7, 345_8 и 345_9 в десятичную систему.
5. Переведите число 194 в троичную, шестеричную, семеричную и восьмеричную системы счисления.
6. Какие из чисел 1230_7, 124_7, 600_7, 530_7 делятся на 7? На 49?
7. Десятичное число, переведённое в восьмеричную и в девятеричную систему, в обоих случаях заканчивается на цифру 0. Какое минимальное десятичное число удовлетворяет этому условию?
8. Сколько всего раз встречается цифра 2 в записи чисел 10, 11, 12, ..., 17 в системе счисления с основанием 5?
9. Сколько всего раз встречается цифра 3 в записи чисел 19, 20, 21, ..., 33 в системе счисления с основанием 6?

10. В системе счисления с некоторым основанием x число 12 записывается в виде 110_x. Найдите это основание.

11. Найдите все основания систем счисления, в которых запись числа 29 оканчивается на 5.

12. В системе счисления с некоторым основанием N десятичное число 129 записывается как 1004_N. Найдите это основание.

13. Запись числа 30 в системе счисления с основанием N выглядит так: 110_N. Укажите основание N этой системы счисления.

14. Запись числа 23 в системе счисления с основанием N выглядит так: 212_N. Укажите основание N этой системы счисления.

15. Запись числа 210_5 в системе счисления с основанием N выглядит так: 313_N. Укажите основание N этой системы счисления.

16. Запись числа 65_8 в системе счисления с основанием N выглядит так: 311_N. Укажите основание N этой системы счисления.

17. Найдите все десятичные числа, не превосходящие 25, запись которых в двоичной системе счисления оканчивается на 101.

18. Запись числа 67 в системе счисления с основанием N оканчивается на 1 и содержит 4 цифры. Укажите основание N этой системы счисления.

19. Найдите все основания систем счисления, в которых запись числа 30 оканчивается на 8.

20. Найдите все основания систем счисления, в которых запись числа 31 оканчивается на 11.

21. Найдите все основания систем счисления, в которых запись числа 63 оканчивается на 23.

22. Найдите наименьшее основание системы счисления, в которой запись числа 30 трёхзначна.

23. Найдите наименьшее основание системы счисления, в которой запись числа 70 трёхзначна.

24. Найдите все десятичные числа, не превосходящие 26, запись которых в троичной системе счисления оканчивается на 22.

25. Найдите все десятичные числа, не превосходящие 30, запись которых в четверичной системе счисления оканчивается на 31.

***26.** Найдите все десятичные числа, не превосходящие 25, запись которых в системе счисления с основанием 6 начинается на 4.

***27.** Найдите все десятичные числа, не превосходящие 30, запись которых в системе счисления с основанием 5 начинается на 3.

***28.** Найдите основание системы счисления x, для которого выполняется равенство

а) $32_x + 64_x = 106_x$; б) $45_x + 55_x = 122_x$;

в) $42_x + 41_x = 133_x$; г) $91_x + 93_x = 154_x$.

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

1. ААААА
2. ААААО
3. ААААУ
4. АААОА

Выполните следующие задания:

а) определите, сколько всего слов в списке;

б) укажите слова, которые стоят на 101-м, 125-м, 170-м и 210-м местах;

в) укажите порядковые номера слов ОАОАО, УАУАУ, АОУОА, УОАОУ;

г) укажите номера первого и последнего слов, которые начинаются с буквы О.

30. Все 5-буквенные слова, составленные из букв А, К, Р, У, записаны в алфавитном порядке. Вот начало списка:

1. ААААА
2. ААААК
3. ААААР
4. ААААУ
5. АААКА

Выполните следующие задания:

а) определите, сколько всего слов в списке;

б) укажите слова, которые стоят на 150-м, 250-м, 350-м и 450-м местах;

в) укажите порядковые номера слов АКУРА, КАРАУ, РУКАА, УКАРА, УРАКА;

г) укажите номера первого и последнего слов, которые начинаются с буквы Р.

Дробные числа

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

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

Все множители, на которые умножаются значения цифр, представляют собой отрицательные степени числа 10 — основа-

ния системы счисления. То есть можно использовать развёрнутую форму записи, вводя отрицательные разряды:

разряды → −1 −2 −3 −4

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

Это число можно представить также с помощью схемы Горнера:

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

Рассмотрим дробное число $0,a_1a_2a_3a_4$, записанное в системе счисления с основанием p. Здесь a_1, a_2, a_3, a_4 — это отдельные цифры, стоящие соответственно в разрядах −1, −2, −3 и −4. Это число может быть записано в развёрнутой форме

разряды → −1 −2 −3 −4

$$0,\ a_1\ a_2\ a_3\ a_4 = a_1 \cdot p^{-1} + a_2 \cdot p^{-2} + a_3 \cdot p^{-3} + a_4 \cdot p^{-4}$$

или с помощью схемы Горнера:

$$0,a_1a_2a_2a_4 = p^{-1} \cdot (a_1 + p^{-1} \cdot (a_2 + p^{-1} \cdot (a_3 + p^{-1} \cdot a_4))).$$

Умножив это число на p, получаем $a_1,a_2a_3a_4$. Если взять целую часть результата, мы получим цифру a_1. Таким же способом можно найти оставшиеся цифры дробной части: на каждом шаге умножаем дробную часть на p и запоминаем *целую часть результата* — это и будет очередная цифра записи числа в системе с основанием p. Например, переведём число 0,9376 в пятеричную систему (табл. 2.2).

Таблица 2.2

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

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

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

Вычисления заканчиваются, когда при очередном умножении дробная часть результата равна нулю. Это означает, что все остальные цифры дробной части — нули. В любом ли случае это произойдёт? К сожалению, нет. Чтобы убедиться в этом, вы можете попробовать перевести в пятеричную систему число 0,3 (должна получиться бесконечная дробь). Такая ситуация может случиться в любой системе счисления (например, вспомните, что число $^1/_3$ записывается в виде бесконечной десятичной дроби). В этом случае обычно задают нужное количество значащих цифр и округляют число соответствующим образом.

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

$$25{,}375 = 25 + 0{,}375,$$

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

Вопросы и задания

1. Сформулируйте алгоритм перевода дробной части десятичного числа в шестеричную систему счисления.
2. Как вы думаете, почему не все конечные десятичные дроби можно представить в виде конечных дробей в других системах счисления?
3. Какие дробные десятичные числа можно записать в виде конечной дроби в шестеричной системе счисления? Ответ обоснуйте.

Задачи

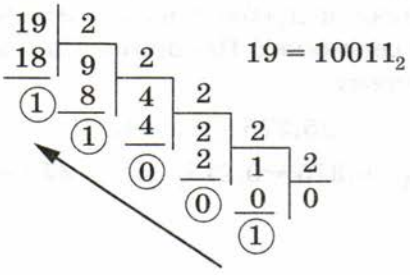
1. Запишите число $0{,}12321_4$ в развёрнутой форме и с помощью схемы Горнера.
2. Переведите число 15,125 в двоичную, четверичную, шестеричную и восьмеричную системы.
3. Какие из этих чисел больше, чем $^1/_2$: $0{,}011_2$; $0{,}12_3$; $0{,}21_4$; $0{,}22_5$; $0{,}25_6$; $0{,}35_7$; $0{,}35_8$?
4. Переведите числа 11,125; 15,75; 22,6875 и 30,375 в систему счисления с основанием 4.

§ 11
Двоичная система счисления

Основные понятия

В **двоичной системе счисления**, т. е. в системе с основанием 2, алфавит состоит из двух цифр: 0 и 1. Все данные в компьютерных устройствах хранятся и обрабатываются как числа, представленные в двоичной системе счисления.

Для перевода натуральных чисел из десятичной системы в двоичную можно использовать общий алгоритм, описанный в предыдущем параграфе (деление на 2 и выписывание остатков в обратном порядке). Например, переведём в двоичную систему число 19:



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

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

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

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

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

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

Мы разложили число на сумму степеней двойки. Для «полного комплекта» здесь не хватает 2^5, 2^4 и 2^1, но можно считать, что эти степени умножаются на нули:

$$77 = (1)\cdot 2^6 + (0)\cdot 2^5 + (0)\cdot 2^4 + (1)\cdot 2^3 + (1)\cdot 2^2 + (0)\cdot 2^1 + (1)\cdot 2^0$$

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

$$\begin{array}{r} 6\,5\,4\,3\,2\,1\,0 \leftarrow \text{разряды} \\ 77 = 1001101_2. \end{array}$$

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

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

Кроме того, иногда удобно применять схему Горнера. В первом столбце таблицы записывают цифры в разрядах двоичного числа, начиная со старшей. Вычисления начинаются с 1 (старший разряд всегда равен 1, если число — не ноль). В каждой из следующих строчек результат, полученный в предыдущей строчке, умножается на 2, и к нему прибавляется очередная цифра двоичного числа (из первой ячейки той же строки) (табл. 2.3).

Таблица 2.3

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

Арифметические операции

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

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

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

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

$$\begin{array}{r} 11111 \\ 10110_2 \\ +\ 111011_2 \\ \hline 1010001_2 \end{array}$$

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

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

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

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

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

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

$$\begin{array}{r} \dot{1}10 \\ -\,\cancel{2}1 \\ 9 \\ \hline 12 \end{array}$$

Здесь точкой сверху обозначен разряд, из которого берётся заём.

Более сложный случай — заём из дальнего (не ближайшего) разряда. Вычтем 9 из 2001. В этом случае занять из ближайшего разряда не удаётся (там 0), поэтому берем заём из того разряда, где стоит цифра 2. Все промежуточные разряды в результате за-

полняются цифрой 9, это старшая цифра десятичной системы счисления:

$$\begin{array}{r} 19910 \\ -\ \ 9 \\ \hline 1992 \end{array}$$

Что изменится в двоичной системе? Когда берется заём, в «рабочий» разряд добавляется уже не 10, а $10_2 = 2$ (основание системы счисления), а все «промежуточные» разряды (между «рабочим» и тем, откуда берётся заём) заполняются не девятками, а единицами (старшей цифрой системы счисления). Например:

$$\begin{array}{r} 10000_2 \\ -\ \ 1_2 \\ \hline 10111_2 \end{array} \qquad \begin{array}{r} 1000101_2 \\ -\ \ 10111_2 \\ \hline 101010_2 \end{array}$$

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

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

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

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

Дробные числа

Для перевода дробного числа в двоичную систему используется общий подход, описанный в § 10. В данном случае нужно умножать число на 2, запоминать целую часть и отбрасывать её

перед следующим умножением. Например, для числа 0,8125 получаем табл. 2.4.

Таблица 2.4

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

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

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

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

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



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

Отметим, что эта проблема связана не с двоичной системой, а с ограниченным размером ячейки памяти компьютера, отведённой на хранение числа. В любой системе счисления сущест-

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

вуют бесконечные дроби, которые не могут быть точно представлены конечным числом разрядов.

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

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

Выводы

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

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

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

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

Вопросы и задания

1. Как вы думаете, какие дробные числа могут быть точно представлены в памяти компьютера в двоичном коде?
2. Почему рекомендуется выполнять вычисления, используя только операции с целыми числами, если есть такая возможность?
3. Как можно работать с дробными числами, не теряя в точности? В чём *недостатки* такого подхода?

Подготовьте сообщение

«Двоичная система счисления с точки зрения человека и компьютера»

Задачи

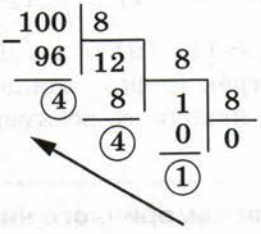
1. Переведите числа 25, 31, 37, 63, 85, 127, 128 в двоичную систему счисления.
2. Переведите числа 100011_2, 101101_2, 110111_2, 1001011_2, 1011111_2, 1101001_2 в десятичную систему счисления.
3. Сколько единиц в двоичной записи чисел 173, 195, 126, 208?
4. Сколько значащих нулей в двоичной записи чисел 48, 73, 96, 254?
5. Как по записи числа в двоичной системе счисления определить, что оно чётное? Делится на 4? Делится на 8? Делится на 32?
6. Выполните сложение в двоичной системе:
   а) $1010111_2 + 110101_2$;
   б) $1010111_2 + 110101_2$;
   в) $101101_2 + 11111_2$;
   г) $10111_2 + 101110_2$;
   д) $111011_2 + 11011_2$;
   е) $111011_2 + 10011_2$.
   Для проверки повторите вычисления, переходя к десятичной системе, а потом преобразуя результат обратно в двоичную.
7. Выполните вычитание в двоичной системе:
   а) $101101_2 - 11111_2$;
   б) $11011_2 - 110101_2$;
   в) $10111_2 - 101110_2$;
   г) $101011_2 - 11011_2$;
   д) $1011_2 - 100101_2$;
   е) $1001_2 - 101101_2$.
   Для проверки повторите вычисления, переходя к десятичной системе, а потом преобразуя результат обратно в двоичную.

8. Переведите в двоичную систему числа 13,125; 23,25; 37,375; 48,625; 78,875.
9. Переведите в двоичную систему числа 11,8; 15,3; 22,7, выделив период в дробной части.
10. Требуется определить, верно ли, что среднее арифметическое 100 целых чисел превышает 0,2. Как сделать это, не используя операции с дробными числами?

§ 12
Восьмеричная система счисления

Восьмеричная система счисления (система с основанием 8) использовалась для кодирования команд во многих компьютерах 1950–1980-х гг. (например, в американской серии PDP-11, советских компьютерах серий ДВК, СМ ЭВМ, БЭСМ). В ней используются цифры от 0 до 7.

Для перевода десятичного числа в восьмеричную систему проще всего использовать стандартный алгоритм для позиционных систем (деление на 8, выписывание остатков в обратном порядке). Например:



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

разряды → 2 1 0

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

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

Оказывается, можно сделать перевод из восьмеричной системы в двоичную напрямую, используя тесную связь между этими системами: их основания связаны равенством $2^3 = 8$. Покажем это на примере восьмеричного числа 753_8. Запишем его в развёрнутой форме:

$$753_8 = 7 \cdot 8^2 + 5 \cdot 8^1 + 3 \cdot 8^0 = 7 \cdot 2^6 + 5 \cdot 2^3 + 3 \cdot 2^0.$$

Теперь переведём отдельно каждую цифру в двоичную систему:

$$7 = 111_2 = 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0, \quad 5 = 101_2 = 1 \cdot 2^2 + 1 \cdot 2^0,$$
$$3 = 11_2 = 1 \cdot 2^1 + 1 \cdot 2^0.$$

Подставим эти выражения в предыдущее равенство:

$$753_8 = (1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0) \cdot 2^6 + (1 \cdot 2^2 + 1 \cdot 2^0) \cdot 2^3 + (1 \cdot 2^1 + 1 \cdot 2^0) \cdot 2^0.$$

Раскрывая скобки, мы получим разложение исходного числа по степеням двойки, т. е. его запись в двоичной системе счисления (здесь добавлены нулевые слагаемые для отсутствующих степеней числа 2):

$$753_8 = ①\cdot 2^8 + ①\cdot 2^7 + ①\cdot 2^6 + ①\cdot 2^5 + ⓪\cdot 2^4 +$$
$$+ ①\cdot 2^3 + ⓪\cdot 2^2 + ①\cdot 2^1 + ①\cdot 2^0.$$

Таким образом, $753_8 = 111\ 101\ 011_2$. Двоичная запись разбита на *триады* (группы из трёх цифр), каждая триада — это двоичная запись одной цифры исходного восьмеричного числа.

Алгоритм перевода восьмеричного числа в двоичную систему счисления:

1. Перевести значение каждой цифры (отдельно) в двоичную систему. Записать результат в виде триады, добавив, если нужно, нули в начало (табл. 2.5).
2. Соединить триады в одно «длинное» двоичное число.

Например: $35721_8 = 11\ 101\ 111\ 010\ 001_2$. В этой записи триады специально отделены друг от друга пробелом. Обратите внимание, что все триады дополнены слева нулями до трёх цифр:

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

Таблица 2.5

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

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

Алгоритм перевода двоичного числа в восьмеричную систему счисления:

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

Например, переведём в восьмеричную систему число 1010011100101110111_2. Разобьём его на триады (начиная справа), к первой триаде нужно добавить два нуля (они подчёркнуты):

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

Далее по табл. 2.5 переводим каждую триаду в восьмеричную систему:

$$1010011100101110111_2 = 1234567_8.$$

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

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

При вычислениях в восьмеричной системе нужно помнить, что максимальная цифра — это 7. Перенос при сложении возникает тогда, когда сумма в очередном разряде получается больше 7. Заём из старшего разряда равен $10_8 = 8$, а все «промежуточные» разряды заполняются цифрой 7 — старшей цифрой системы счисления. Приведём примеры сложения и вычитания:

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

$$\begin{array}{rl} \ {\scriptstyle \cdot\cdot} & \\ -\ 456_8 & (6+8)-7=\textcircled{7} \\ \hline 277_8 & (5-1+8)-7=\textcircled{5} \\ 157_8 & (4-1)-2=\textcircled{1} \end{array}$$

В примере на сложение запись 1 · 8 + 2 означает, что получилась сумма, бóльшая 7, которая не помещается в один разряд. Единица идёт в перенос, а двойка остаётся в этом разряде. В записи операций при выполнении вычитания запись «−1» означает, что из этого разряда раньше был заём (его значение уменьшилось на 1), а запись «+8» означает заём из старшего разряда.

С помощью восьмеричной системы удобно кратко записывать содержимое областей памяти, содержащих количество битов, кратное трём. Например, 6-битные данные «упаковываются» в две восьмеричные цифры. Некоторые компьютеры 1960-х годов использовали 24-битные и 36-битные данные, они записывались соответственно с помощью 8 и 12 восьмеричных цифр. Восьмеричная система использовалась даже для компьютеров с 8-битной ячейкой памяти (PDP-11, ДВК), но позднее была почти вытеснена шестнадцатеричной системой (см. далее).

Сейчас восьмеричная система применяется, например, для установки прав на доступ к файлу в операционной системе Linux (и других Unix-системах) с помощью команды `chmod`. Ре-

жим доступа кодируется тремя битами, которые разрешают чтение (r, read, старший бит), запись (w, write) и выполнение файла (x, execute, младший бит). Код $7 = 111_2$ (rwx) означает, что все биты установлены (полный доступ), а код $5 = 101_2$ (r-x) разрешает чтение и выполнение файла, но запрещает его изменение.

Вопросы и задания

1. Какие цифры составляют алфавит восьмеричной системы счисления? Обоснуйте ответ.
2. Сколько существует различных двузначных восьмеричных чисел? Приведите примеры.

Задачи

1. Переведите числа 49, 53, 64, 150, 266 в восьмеричную и двоичную системы счисления.
2. Переведите числа 123_8, 234_8, 345_8, 456_8 и 567_8 в десятичную и двоичную системы счисления.
3. Запишите числа 101111001_2, 10110100_2, 1000011_2, 10101010_2 в восьмеричной и десятичной системах счисления.
4. Вычислите значения следующих выражений:

   а) $353_8 + 736_8$;
   б) $1353_8 + 777_8$;
   в) $1153_8 - 662_8$;
   г) $153_8 - 662_8$.

5. Вычислите значения следующих выражений, запишите результат в двоичной, восьмеричной и десятичной системах счисления:

   а) $45_8 + 1010110_2$;
   б) $271_8 + 11110100_2$;
   в) $110111_2 + 135_8$;
   г) $10 + 10_8 \cdot 10_2$;
   д) $123 + 12_8 \cdot 11_2$;
   е) $153_8 - 16 \cdot 101_2$;
   ж) $15_8 \cdot 110_2$;
   з) $50_8 \cdot 21_8$;
   и) $34_8 : 10101_2$;
   к) $214_8 : 1110_2$.

*6. Переведите число 12,5 в восьмеричную систему счисления.

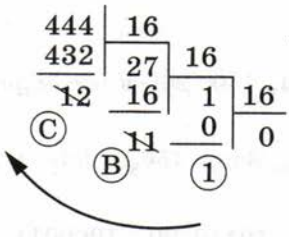
§ 13
Шестнадцатеричная система счисления

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

$$A = 10, \quad B = 11, \quad C = 12, \quad D = 13, \quad E = 14, \quad F = 15.$$

Таким образом, старшая цифра в шестнадцатеричной системе — F.

Для перевода чисел из десятичной системы в шестнадцатеричную используют алгоритм деления на 16 и взятия остатков. Важно не забыть, что все остатки, бóльшие 9, нужно заменить на буквы:



$444 = 1BC_{16}$

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

разряды → 2 1 0

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

Можно также использовать схему Горнера:

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

Основания двоичной и шестнадцатеричной систем связаны соотношением $2^4 = 16$, поэтому можно переводить числа из шестнадцатеричной системы в двоичную напрямую. Алгоритмы перевода чисел из шестнадцатеричной системы в двоичную и обратно полностью аналогичны соответствующим алгоритмам для восьмеричной системы. Каждая шестнадцатеричная цифра представляется в виде *тетрады* (группы из четырёх двоичных цифр) (табл. 2.6).

Таблица 2.6

0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111

8	1000
9	1001
A (10)	1010
B (11)	1011
C (12)	1100
D (13)	1101
E (14)	1110
F (15)	1111

Алгоритм перевода шестнадцатеричного числа в двоичную систему счисления:

1. Перевести значение каждой цифры (отдельно) в двоичную систему. Записать результат в виде тетрады, добавив, если нужно, нули в начало (см. табл. 2.6).
2. Соединить тетрады в одно «длинное» двоичное число.

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

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

Обратите внимание, что для цифр, меньших 8 (кроме первой), результат перевода в двоичную систему нужно дополнить старшими нулями до 4 знаков.

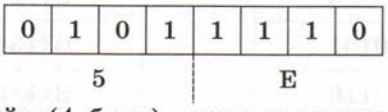
Алгоритм перевода двоичного числа в шестнадцатеричную систему счисления:

1. Разбить двоичное число на тетрады, *начиная справа*. В начало самой первой тетрады добавить слева нули, если это необходимо.
2. Перевести каждую тетраду (отдельно) в шестнадцатеричную систему счисления.
3. Соединить полученные цифры в одно «длинное» число.

Например:

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

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



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

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

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

$A5B_{16}$	1 1	$11 + 14 = 1 \cdot 16 + ⑨$
$+\ C7E_{16}$	$+\ 10\ 5\ 11$	$5 + 7 + 1 = 13 = Ⓓ$
$16D9_{16}$	$12\ 7\ 14$	$10 + 12 = 1 \cdot 16 + ⑥$
	$1\ 6\ 13\ 9$	$0 + 0 + 1 = ①$

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

$C5B_{16}$	$\dot{1}\dot{2}\ \dot{5}\ 11$	$(11 + 16) - 14 = 13 = Ⓓ$
$+\ A7E_{16}$	$-\ 10\ 7\ 14$	$(5 - 1 + 16) = -7 = 13 = Ⓓ$
$1DD_{16}$	$1\ 13\ 13$	$(12 - 1) - 10 = ①$

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

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

$$53_8 = 101\ 011_2 = 10\ 1011_2 = 2B_{16}.$$

Теперь сложим числа в 16-ричной системе:

$$2B_{16} + 56_{16} = 81_{16}$$

и переведём результат в двоичную систему:

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

Вопросы и задания

1. Какие цифры используются в шестнадцатеричной системе? Сколько их?
2. Почему появилась необходимость использовать латинские буквы?
3. Сформулируйте алгоритмы перевода чисел из шестнадцатеричной системы счисления в двоичную и обратно.
4. Какое минимальное основание должно быть у системы счисления, чтобы в ней могли быть записаны числа 123, 4AB, 9A3 и 8455?

Задачи

1. Переведите в двоичную и восьмеричную системы числа $7F1A_{16}$, $C73B_{16}$, $2FE1_{16}$, $A112_{16}$.
2. Переведите в двоичную и шестнадцатеричную системы числа 6172_8, 5341_8, 7711_8, 1234_8.
3. Переведите в восьмеричную и шестнадцатеричную системы числа
   а) 1110111101010_2;
   б) 1010101101010110_2;
   в) 111100110111110101_2;
   г) 110110110101111110_2.

4. Переведите числа 29, 43, 54, 120, 206 в шестнадцатеричную, восьмеричную и двоичную системы счисления.

5. Переведите числа 73_8, 134_8, 245_8, 356_8 и 467_8 в шестнадцатеричную, десятичную и двоичную системы счисления.

6. Запишите числа 10110101_2, 1110100_2, 1000111_2, 10111110_2 в шестнадцатеричной, восьмеричной и десятичной системах счисления.

7. Вычислите значения следующих выражений:

а) $3AF_{16} + 1CBE_{16}$;
б) $1EA_{16} + 7D7_{16}$;
в) $A81_{16} + 377_{16}$;
г) $1CFB_{16} - 22F_{16}$;
д) $22F_{16} - CFB_{16}$;
е) $1AB_{16} - 2CD_{16}$.

8. Вычислите значения следующих выражений, запишите результат в двоичной, восьмеричной, десятичной и шестнадцатеричной системах счисления:

а) $4F_{16} + 111110_2$;
б) $5A_{16} + 1010111_2$;
в) $256_8 + 2C_{16}$;
г) $110111_2 + 135_8$;
д) $12_{16} + 12_8 \cdot 11_2$;
е) $35_8 + 2C_{16} \cdot 101_2$.

9. Вычислите значения следующих выражений, запишите результат в двоичной, восьмеричной, десятичной и шестнадцатеричной системах счисления:

а) $15_{16} \cdot 110_2$;
б) $2A_{16} \cdot 12_8$;
в) $34_{16} : 32_8$;
г) $740_8 : 181_6$.

*10. Переведите числа 49,6875 и 52,9 в шестнадцатеричную систему счисления.

§ 14
Другие системы счисления

Троичная уравновешенная система счисления

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

В троичной уравновешенной системе основание равно 3, используются три цифры: $\bar{1}$ («минус 1»), 0 и 1. Один троичный разряд называется **тритом** (в отличие от двоичного бита). Система называется **уравновешенной**, потому что с помощью любого числа разрядов можно закодировать равное число положительных и отрицательных чисел, и число ноль. В таблице 2.7 показаны, например, все двухразрядные числа.

Таблица 2.7

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

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

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

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

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

Каждая гиря может быть в трёх состояниях:

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

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

Двоично-десятичная система счисления

Существует ещё один простой способ записи десятичных чисел с помощью цифр 0 и 1. Этот способ называется **двоично-десятичной системой** (ДДС), это нечто среднее между двоичной и десятичной системами. На английском языке такое кодирование называется *binary coded decimal* (BCD) — десятичные числа, закодированные двоичными цифрами.

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

$$\underset{9}{1001}\ \underset{0}{0000}\ \underset{2}{0010}\ \underset{4}{0100},\ \underset{1}{0001}\ \underset{9}{1001}_{ДДС}$$

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

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

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

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

Использование ДДС дает следующие **преимущества**:

- двоично-десятичный код очень легко переводить в десятичный, например, для вывода результата на экран;
- просто выполняется умножение и деление на 10, а также округление;
- конечные десятичные дроби записываются точно, без ошибки, так что вычисления в ДДС (вместо двоичной системы)

дадут тот же результат, что и ручные расчёты человека «на бумажке»; поэтому ДДС используется в калькуляторах.

Есть, однако, и ***недостатки***:

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

Вопросы и задания

1. Как поменять знак числа, записанного в уравновешенной системе счисления?
2. Сколько положительных и отрицательных чисел можно закодировать с помощью 5 разрядов в троичной уравновешенной системе счисления?

*3. Попробуйте сформулировать правила сложения чисел в троичной уравновешенной системе.

*4. Сравните троичную уравновешенную систему счисления с двоичной. Как вы думаете, почему разработчики компьютеров все-таки выбрали двоичный код для хранения данных?

*5. Верно ли, что любая последовательность нулей и единиц может быть числом, закодированным в двоично-десятичной системе? Обоснуйте свой ответ.

6. Какие числа записываются одинаково в двоичной и двоично-десятичной системах счисления?

Подготовьте сообщение

«Сравнение двоичной и двоично-десятичной систем счисления»

Задачи

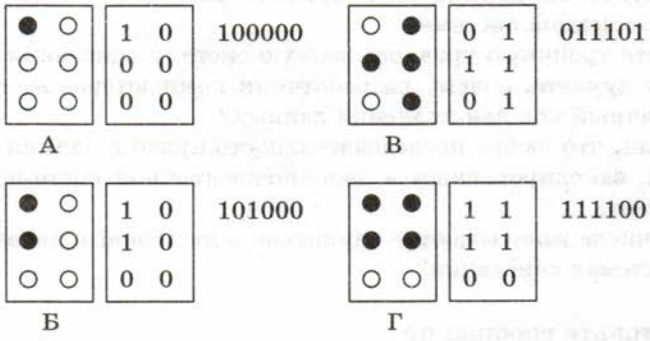
1. Запишите числа –15 и 15 в троичной уравновешенной системе. Сколько разрядов вам потребовалось?
2. Найдите минимальный набор гирь, с помощью которых на чашах равноплечных весов можно было взвесить груз массой от 1 до 13 кг включительно.
3. Закодируйте число 1234 в двоично-десятичной системе счисления.
4. Запишите число $1011110000110100 1_{ДДС}$ в десятичной системе счисления.

*5. Найдите в литературе или в Интернете материалы о факториальной и фибоначчиевой системах счисления.

§ 15 Кодирование символов

Общий подход

Поскольку в современных компьютерах информация всех видов представлена в двоичном коде, нужно разобраться, как закодировать символы в виде цепочек нулей и единиц. Например, можно предложить способ, основанный на системе Брайля для незрячих людей, о которой мы уже упоминали (см. задачу 17 на стр. 66). В нём каждый символ кодируется с помощью 6 точек, расположенных в два столбца. В каждой точке может быть выпуклость, которая чувствуется на ощупь. Обозначив выпуклость единицей, а её отсутствие — нулём, можно закодировать первые буквы русского алфавита следующим образом:



Здесь двоичный код строится так: строки полученной таблицы, состоящей из цифр 0 и 1, выписываются одна за другой в строчку. Так как используются всего 6 точек, количество символов, которые можно закодировать, равно $2^6 = 64$ (в реальной системе Брайля 63 символа, потому что символ, в коде которого нет ни одной выпуклости, невозможно обнаружить на ощупь).

Понятно, что совершенно не обязательно использовать код Брайля. Главное — каждому используемому символу как-то сопоставить цепочку нулей и единиц, например составить таблицу «символ — код». На практике поступают следующим образом:

1) определяют, сколько символов нужно использовать (обозначим это число через N);
2) определяют нужное количество k двоичных разрядов так, чтобы с их помощью можно было закодировать не менее N разных последовательностей (т. е. $2^k \geq N$);

3) составляют таблицу, в которой каждому символу сопоставляют код (номер) — целое число в интервале от 0 до $2^k - 1$;
4) коды символов переводят в двоичную систему счисления.

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

Кодировка ASCII и её расширения

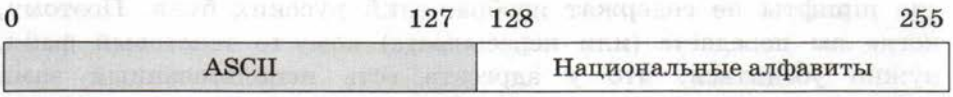
Для того чтобы упростить передачу текстовой информации, разработаны стандарты, которые закрепляют определённые коды за общеупотребительными символами. Основным международным стандартом является 7-битная **кодировка ASCII** (англ. *American Standard Code for Information Interchange* — американский стандартный код для обмена информацией), в которую входят $2^7 = 128$ символов с кодами от 0 до 127:

- служебные (управляющие) символы с кодами от 0 до 31;
- символ «пробел» с кодом 32;
- цифры от «0» до «9» с кодами от 48 до 57;
- латинские буквы: заглавные, от «A» до «Z» (с кодами от 65 до 90) и строчные, от «a» до «z» (с кодами от 97 до 122);
- знаки препинания: . , : ; ! ?
- скобки: [] {} ()

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

- математические символы: + – * / = < >
- некоторые другие знаки: " ' # $ % & ^ | @ \ _ ~

В современных компьютерах минимальная единица памяти, имеющая собственный адрес, — это байт (8 битов). Поэтому для хранения кодов ASCII в памяти можно добавить к ним ещё один (старший) нулевой бит, таким образом, получая 8-битную кодировку. Кроме того, дополнительный бит можно использовать: он даёт возможность добавить в таблицу еще 128 символов с кодами от 128 до 255. Такое расширение ASCII часто называют **кодовой страницей**. Первую половину кодовой страницы (коды от 0 до 127) занимает стандартная таблица ASCII, а вторую — символы национальных алфавитов (например, русские буквы):



Кодовая страница

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

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

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

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

Для веб-страниц в Интернете часто используют кодировки Windows-1251 и KOI8-R. Браузер после загрузки страницы пытается автоматически определить ее кодировку. Если ему это не удаётся, вы видите странный набор букв вместо понятного русско-

го текста. В этом случае нужно сменить кодировку вручную с помощью меню **Вид** браузера.

Стандарт UNICODE

Любая 8-битная кодовая страница имеет серьёзное ограничение — она может включать только 256 символов. Поэтому не получится набрать в одном документе часть текста на русском языке, а часть — на китайском. Кроме того, существует проблема чтения документов, набранных с использованием другой кодовой страницы. Всё это привело к принятию в 1991 г. нового стандарта кодирования символов — **UNICODE**, который позволяет записывать знаки любых существующих (и даже некоторых «мёртвых») языков, математические и музыкальные символы и др.

Если мы хотим расширить количество используемых знаков, необходимо увеличивать место, которое отводится под каждый символ. Вы знаете, что компьютер работает сразу с одним или несколькими байтами, прочитанными из памяти. Например, если увеличить место, отводимое на каждый символ, до двух байтов, то можно закодировать $2^{16} = 65\ 536$ символов в одном наборе. В современной версии UNICODE можно кодировать до 1 112 064 различных знаков, однако реально используются немногим более 100 000 символов.

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

В Unix-подобных системах, например в Linux, чаще применяют кодировку **UTF-8**. В ней все символы, входящие в таблицу ASCII, кодируются в виде 1 байта, а другие символы могут занимать от 2 до 4 байтов. Если значительную часть текста составляют латинские буквы и цифры, такой подход позволяет значительно уменьшить объём файла по сравнению с UTF-16. Текст, состоящий только из символов таблицы ASCII, кодируется точно так же, как и в кодировке ASCII. По данным поисковой системы Google, на начало 2010 г. около 50% сайтов в Интернете использовали кодировку UTF-8.

Кодировки стандарта UNICODE позволяют использовать символы разных языков в одном документе. За это приходится «расплачиваться» увеличением объёма файлов.

Вопросы и задания

1. Какая информация хранится в текстовом файле?
2. Что такое шрифтовой файл?
3. Вы хотите использовать в тексте придуманный собственный символ, которого нет ни в одном шрифте. Какими путями это можно сделать?
4. Вы сами разработали шрифт и хотите переслать другу документ, в котором этот шрифт используется. Какими способами это можно сделать?
5. В чём *недостатки* и преимущества внедрения шрифтов в документ?
6. Что представляет собой кодировка ASCII? Сколько символов включает эта кодировка?
7. Почему в современных компьютерах используются кодировки, в которых каждый символ занимает целое число байтов?
8. Что такое кодовая страница?
9. Назовите основные кодовые станицы, содержащие русские буквы.
10. Почему использование кодовых страниц для кодирования текста может привести к проблемам?
11. Что делать, если вы видите непонятный набор символов на веб-странице?
12. В чём состоит ограничение 8-битных кодировок?

Подготовьте сообщение

а) «Стандарт UNICODE: за и против»
б) «Кодировка UTF-16»
в) «Кодировка UTF-8»

Задачи

1. Сколько символов можно закодировать с помощью 5-битного кода? 9-битного?
2. Сколько битов нужно выделить на символ для того, чтобы использовать в одном документе 100 разных символов? 200? 500?
3. Какой символ имеет код 100 в кодировке ASCII?
4. Какой код имеет цифра «5» в кодировке ASCII?
5. Определите, чему равен информационный объём следующего высказывания Рене Декарта, закодированного с помощью 16-битной кодировки UNICODE:

   `Я мыслю, следовательно, существую.`
6. При перекодировке сообщения на русском языке из 16-битного кода UNICODE в 8-битную кодировку KOI8-R оно уменьшилось на 480 битов. Какова длина сообщения в символах?

7. При перекодировке сообщения из 8-битного кода в 16-битную кодировку UNICODE его объём увеличился на 2048 байтов. Каков был информационный объём сообщения до перекодировки?
8. В таблице представлена часть кодовой таблицы ASCII:

Символ	1	5	A	B	Q	a	b
Десятичный код	49	53	65	66	81	97	98
Шестнадцатеричный код	31	35	41	42	51	61	62

Каков шестнадцатеричный код символа «q»?

§ 16
Кодирование графической информации

Как и все виды информации, изображения в компьютере закодированы в виде двоичных последовательностей. Используют два принципиально разных метода кодирования графической информации, каждый из которых имеет свои *достоинства* и *недостатки*.

Растровое кодирование

Рисунок состоит из линий и закрашенных областей. В идеале нам нужно закодировать все особенности этого изображения так, чтобы его можно быть в точности восстановить из кода (например, распечатать на принтере).

И линия, и область состоят из бесконечного числа точек. Цвет каждой из этих точек нам нужно закодировать. Если их бесконечно много, мы сразу приходим к выводу, что для этого нужно бесконечно много памяти. Поэтому «поточечным» способом изображение закодировать не удастся. Однако эту идею всё-таки можно использовать.

Начнём с чёрно-белого рисунка. Представим себе, что на изображение ромба наложена сетка, которая разбивает его на квадратики. Такая сетка называется **растром**. Теперь для каждого квадратика определим цвет (чёрный или белый). Для тех квадратиков, в которых часть оказалась закрашена чёрным цветом, а часть — белым, выберем цвет в зависимости от того, какая часть (чёрная или белая) больше (рис. 2.15).

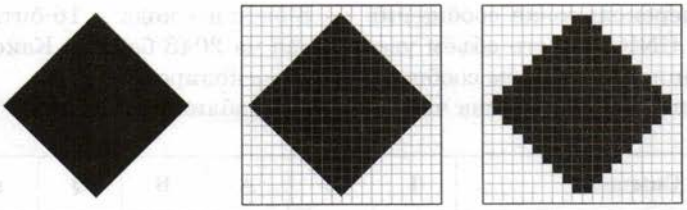


Рис. 2.15

У нас получился так называемый **растровый рисунок**, состоящий из квадратиков-пикселей.

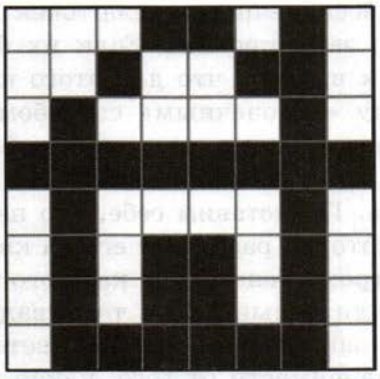
Пиксель (англ. *pixel* — picture element, элемент рисунка) — это наименьший элемент рисунка, для которого можно независимым образом задать свой цвет.

Разбив «обычный» рисунок на квадратики, мы выполнили его **дискретизацию** — разделили единый объект на отдельные элементы. Действительно, у нас был единый рисунок — изображение ромба. В результате мы получили дискретный объект — набор пикселей.

Двоичный код для чёрно-белого рисунка, полученного в результате дискретизации, можно построить следующим образом:

- заменяем белые пиксели нулями, а чёрные — единицами;
- выписываем строки таблицы одну за другой.

Покажем это на простом примере (рис. 2.16).



0	0	0	1	1	0	1	0
0	0	1	0	0	1	1	0
0	1	0	0	0	0	1	0
1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	0
0	1	0	1	1	0	1	0
0	1	0	1	1	0	1	0
0	1	1	1	1	1	1	0

Рис. 2.16

Ширина этого рисунка — 8 пикселей, поэтому каждая строка таблицы состоит из 8 двоичных разрядов — битов. Чтобы не писать очень длинную цепочку нулей и единиц, удобно использовать шестнадцатеричную систему счисления, закодировав 4 соседних бита (тетраду) одной шестнадцатеричной цифрой. Например, для первой строки получаем код $1A_{16}$:

0	0	0	1	1	0	1	0
			1_{16}			A_{16}	

а для всего рисунка: $1A2642FF425A5A7E_{16}$.

Очень важно понять, что мы приобрели и что потеряли в результате дискретизации. Самое важное: мы смогли закодировать рисунок в двоичном коде. Однако при этом рисунок исказился — вместо ромба мы получили набор квадратиков. Причина искажения в том, что в некоторых квадратиках части исходного рисунка были закрашены разными цветами, а в закодированном изображении каждый пиксель обязательно имеет один цвет. Таким образом, часть исходной информации при кодировании была потеряна. Это наглядно проявится, например, при увеличении рисунка — квадратики увеличатся, и рисунок ещё больше исказится. Чтобы уменьшить потери информации, нужно уменьшать размер пикселя, т. е. увеличивать разрешение.

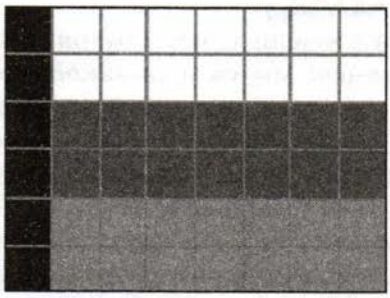
Разрешение — это количество пикселей, приходящихся на единицу линейного размера изображения.

Разрешение обычно измеряется в пикселях на дюйм. Используется английское обозначение ppi — pixels per inch. Например, разрешение 254 ppi означает, что на дюйм (25,4 мм) приходится 254 пикселя, так что каждый пиксель «содержит» квадрат исходного изображения размером 0,1 × 0,1 мм. Если провести дискретизацию рисунка размером 10 × 15 см с разрешением 254 ppi, высота закодированного изображения будет 100/0,1 = 1000 пикселей, а ширина — 1500 пикселей.

Чем больше разрешение, тем точнее кодируется рисунок (меньше информации теряется), однако одновременно растёт и объём файла.

Кодирование цвета

Что делать, если рисунок цветной? В этом случае для кодирования цвета пикселя уже не обойтись одним битом. Например, в показанном на рис. 2.17, *а* (см. также цветной рисунок на форзаце) изображении российского флага 4 цвета: чёрный, синий, красный и белый. Для кодирования одного из четырёх вариантов нужно 2 бита, поэтому код каждого цвета (и код каждого пикселя) будет состоять из двух битов. Пусть 00 обозначает чёрный цвет, 01 — красный, 10 — синий и 11 — белый. Получаем таблицу (рис. 2.17, *б*).



а

00	11	11	11	11	11	11	11
00	11	11	11	11	11	11	11
00	10	10	10	10	10	10	10
00	10	10	10	10	10	10	10
00	01	01	01	01	01	01	01
00	01	01	01	01	01	01	01

б

Рис. 2.17

Проблема только в том, что при выводе на экран нужно как-то определить, какой цвет соответствует тому или другому коду. То есть информацию о цвете для вывода на экран нужно выразить в виде числа (или набора чисел).

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

Согласно современному представлению о **цветном зрении** (теории Юнга—Гельмгольца), глаз человека содержит чувствительные элементы (рецепторы) трёх типов. Каждый из них воспринимает весь поток света, но первые наиболее чувствительны в области красного цвета, вторые — в области зелёного цвета, а третьи — в области синего цвета. Цвет — это результат возбуждения всех трёх типов рецепторов. Поэтому считается, что любой цвет (т. е. ощущения человека, воспринимающего волны опре-

делённой длины) можно имитировать, используя только три световых луча (красный, зелёный и синий) разной яркости. Следовательно, любой цвет (в том числе и «белый») приближённо раскладывается на три составляющих — красную, зелёную и синюю. Меняя силу этих составляющих, можно составить любые цвета (рис. 2.18 и цветной рисунок на форзаце). Эта модель цвета получила название RGB по начальным буквам английских слов «red» (красный), «green» (зелёный) и «blue» (синий).

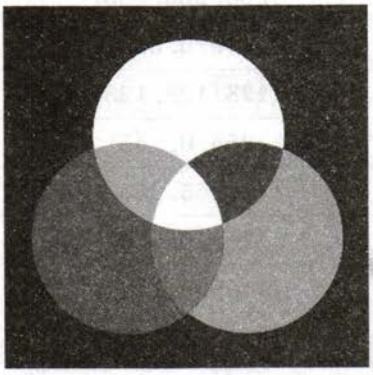


Рис. 2.18

В **модели RGB** яркость каждой составляющей (или, как говорят, каждого канала) чаще всего кодируется целым числом от 0 до 255. При этом код цвета — это тройка чисел (R, G, B) — яркости отдельных каналов. Цвет (0, 0, 0) — это чёрный цвет, а (255, 255, 255) — белый. Если все составляющие имеют равную яркость, получаются оттенки серого цвета: от чёрного до белого.

Чтобы сделать светло-красный (розовый) цвет, нужно при максимальной яркости красного цвета (255, 0, 0) одинаково увеличить яркость зелёного и синего каналов, например, цвет (255, 150, 150) — это розовый. Равномерное уменьшение яркости всех каналов создаёт тёмный цвет, например цвет с кодом (100, 0, 0) — тёмно-красный.

При кодировании цвета на веб-страницах также используется модель RGB, но яркости каналов записываются в шестнадцатеричной системе счисления (от 00_{16} до FF_{16}), а перед кодом цвета ставится знак #. Например, код красного цвета записывается как #FF0000, а код синего — как #0000FF. Коды некоторых цветов приведены в табл. 2.8.

Таблица 2.8

Цвет	Код (R, G, B)	Код на веб-странице
Красный	(255, 0, 0)	#FF0000
Зелёный	(0, 255, 0)	#00FF00
Синий	(0, 0, 255)	#0000FF
Белый	(255, 255, 255)	#FFFFFF
Чёрный	(0, 0, 0)	#000000
Серый	(128, 128, 128)	#808080
Пурпурный[1]	(255, 0, 255)	#FF00FF
Голубой	(0, 255, 255)	#00FFFF
Жёлтый	(255, 255, 0)	#FFFF00
Тёмно-пурпурный	(128, 0, 128)	#800080
Светло-жёлтый	(255, 255, 128)	#FFFF80

Всего есть по 256 вариантов яркости каждого из трёх основных цветов. Это позволяет закодировать $256^3 = 16\ 777\ 216$ оттенков, что более чем достаточно для человека. Так как $256 = 2^8$, каждая из трёх составляющих занимает в памяти 8 битов, или 1 байт, а вся информация о каком-то цвете — 24 бита (3 байта). Эта величина называется глубиной цвета.

Глубина цвета — это количество битов, используемое для кодирования цвета пикселя.

24-битовое кодирование цвета часто называют режимом **истинного цвета** (англ. **True Color** — истинный цвет). Для вычисления объёма рисунка в байтах при таком кодировании нужно определить общее количество пикселей (перемножить ширину и высоту) и умножить результат на 3, так как цвет каждого пикселя кодируется тремя байтами. Например, рисунок размером 20 × 30 пикселей, закодированный в режиме истинного цвета, будет зани-

[1] Пурпурный цвет получается при смешении синего и красного.

мать 20 · 30 · 3 = 1800 байтов. Конечно, здесь не учитывается *сжатие* (уменьшение объёма файлов с помощью специальных алгоритмов), которое применяется во всех современных форматах графических файлов. Кроме того, в реальных файлах есть заголовок, в котором записана служебная информация (например, размеры рисунка).

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

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

Очень часто (например, в схемах, диаграммах и чертежах) количество цветов в изображении невелико (не более 256). В этом случае применяют **кодирование с палитрой**.

Цветовая палитра — это таблица, в которой каждому цвету, заданному в виде составляющих в модели RGB, сопоставляется числовой код.

Кодирование с палитрой выполняется следующим образом:

- выбирается количество цветов N (как правило, не более 256);
- из палитры истинного цвета (16 777 216 цветов) выбираются любые N цветов и для каждого из них находятся составляющие в модели RGB;
- каждому из выбранных цветов присваивается номер (код) от 0 до $N - 1$;
- составляется палитра: сначала записываются RGB-составляющие цвета, имеющего код 0, затем — составляющие цвета с кодом 1 и т. д.;

- цвет каждого пикселя кодируется не в виде значений RGB-составляющих, а как номер цвета в палитре.

Например, при кодировании изображения российского флага (см. выше) были выбраны 4 цвета:

- чёрный: RGB-код (0, 0, 0); двоичный код 00_2;
- красный: RGB-код (255, 0, 0); двоичный код 01_2;
- синий: RGB-код (0, 0, 255); двоичный код 10_2;
- белый: RGB-код (255, 255, 255); двоичный код 11_2;

Поэтому палитра, которая обычно записывается в специальную служебную область в начале файла (эту область называют **заголовком файла**), представляет собой четыре трёхбайтных блока:

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

Код каждого пикселя занимает всего два бита.

Чтобы примерно оценить информационный объём рисунка с палитрой, включающей N цветов, нужно:

- определить размер палитры: $3 \cdot N$ байтов, или $24 \cdot N$ битов;
- определить глубину цвета (количество битов на пиксель), т. е. найти наименьшее натуральное число k, такое что $2^k \geq N$;
- вычислить общее количество пикселей M, перемножив размеры рисунка;
- определить информационный объём рисунка (без учёта палитры): $M \cdot k$ битов.

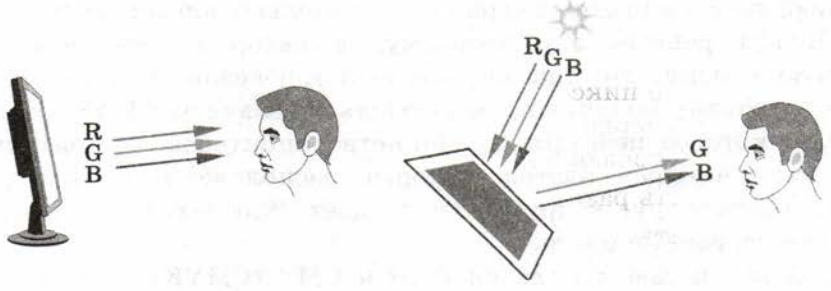
В таблице 2.9 приведены данные по некоторым вариантам кодирования с палитрой.

Таблица 2.9

Количество цветов	Размер палитры, байтов	Глубина цвета, битов на пиксель
2	6	1
4	12	2
16	48	4
256	768	8

Палитры с количеством цветов более 256 на практике не используются.

RGB-кодирование лучше всего описывает цвет, который *излучается* некоторым устройством, например экраном монитора или ноутбука (рис. 2.19, *а* и цветной рисунок на форзаце). Когда же мы смотрим на изображение, отпечатанное на бумаге, ситуация совершенно другая. Мы видим не прямые лучи источника, попадающие в глаз, а *отражённые* от поверхности. «Белый свет» от какого-то источника (солнца, лампочки), содержащий волны во всём видимом диапазоне, попадает на бумагу, на которой нанесена краска. Краска поглощает часть лучей (их энергия уходит на нагрев), а оставшиеся попадают в глаз, это и есть тот цвет, который мы видим (рис. 2.19, *б* и цветной рисунок на форзаце).



а *б*

Рис. 2.19

Например, если краска поглощает красные лучи, остаются только синие и зелёные (см. рис. 2.19, *б*) — мы видим голубой цвет. В этом смысле красный и голубой цвета *дополняют* друг друга, так же как и пары зелёный — пурпурный и синий — жёлтый. Действительно, если из белого цвета (его RGB-код #FFFFFF) «вычесть» зелёный, то получится цвет #FF00FF (пурпурный), а если «вычесть» синий, то получится цвет #FFFF00 (жёлтый).

На трёх дополнительных цветах — голубом, пурпурном и жёлтом — строится цветовая модель CMY (англ. ***Cyan*** — голубой, ***Magenta*** — пурпурный, ***Yellow*** — жёлтый), которая применяется для вывода изображения на печать. Значения C = M = Y = 0 говорят о том, что на белую бумагу не наносится никакая краска, поэтому все лучи отражаются, мы видим белый цвет. Если нанести на бумагу голубой цвет, красные лучи будут поглощаться, оста-

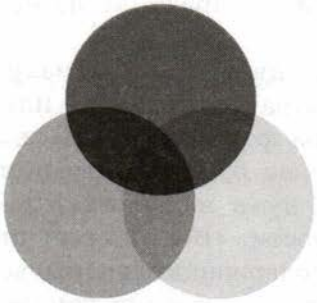


Рис. 2.20

нутся только синие и зелёные. Если сверху нанести ещё жёлтую краску, которая поглощает синие лучи, останется только зелёный цвет (рис. 2.20 и цветной рисунок на форзаце).

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

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

Кроме цветовых моделей RGB и CMY (CMYK) существуют и другие модели. Наиболее интересная из них — **модель HSB**[1] (англ. ***Hue*** — тон, оттенок; ***Saturation*** — насыщенность, ***Brightness*** — яркость), которая ближе всего к естественному восприятию человека. Тон — это, например, синий, зелёный, жёлтый. Насыщенность — это чистота тона, при уменьшении насыщенности до нуля получается серый цвет. Яркость определяет, насколько цвет светлый или тёмный. Любой цвет при снижении яркости до нуля превращается в черный.

Строго говоря, цвет, кодируемый в моделях RGB, CMYK и HSB, зависит от устройства, на котором этот цвет будет изображаться. Для кодирования «абсолютного» цвета применяют **модель Lab** (англ. ***Lightness*** — светлота, **a** и **b** — параметры, определяющие тон и насыщенность цвета), которая является международным стандартом. Эта модель используется, например, для перевода цвета из модели RGB в модель CMYK и обратно.

---

[1] Или **HSV** (англ. ***Hue*** — тон, оттенок; ***Saturation*** — насыщенность, ***Value*** — величина).

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

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



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

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

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

Проблема состоит в том, что не все цвета RGB-модели могут быть напечатаны. В первую очередь это относится к ярким и насыщенным цветам. Например, ярко-красный цвет (R = 255,

G = B = 0) нельзя напечатать, ближайший к нему цвет в модели CMYK (C = 0, M = Y = 255, K = 0) при обратном переводе в RGB может дать значения[1] в районе R = 237, G = 28, B = 26. Поэтому при преобразовании ярких цветов в модель CMYK (и при печати ярких рисунков) они становятся тусклее. Это обязательно должны учитывать профессиональные дизайнеры.

Растровое кодирование: итоги

Итак, при растровом кодировании рисунок разбивается на пиксели (дискретизируется). Для каждого пикселя определяется единый цвет, который чаще всего кодируется с помощью RGB-кода. На практике эти операции выполняет сканер (устройство для ввода изображений) или цифровой фотоаппарат.

Растровое кодирование имеет *достоинства*:

- универсальный метод (можно закодировать любое изображение);
- единственный метод для кодирования и обработки размытых изображений, не имеющих чётких границ, например фотографий;

и *недостатки*:

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

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

[1] Как вы понимаете, точные цифры зависят от профилей монитора и принтера.

теряется. Подробно мы изучим эти вопросы в 11 классе. Чаще всего встречаются следующие форматы файлов:

- **BMP** (англ. *bitmap* — битовая карта; файлы с расширением bmp) — стандартный формат растровых изображений в операционной системе Windows; поддерживает кодирование с палитрой и в режиме истинного цвета;
- **JPEG** (англ. *Joint Photographic Experts Group* — объединенная группа фотографов-экспертов; файлы с расширением jpg или jpeg) — формат, разработанный специально для кодирования фотографий; поддерживает только режим истинного цвета; для уменьшения объёма файла используется сильное сжатие, при котором изображение немного искажается, поэтому не рекомендуется использовать его для рисунков с чёткими границами;
- **GIF** (англ. *Graphics Interchange Format* — формат для обмена изображениями; файлы с расширением gif) — формат, поддерживающий только кодирование с палитрой (от 2 до 256 цветов); в отличие от предыдущих форматов части рисунка могут быть прозрачными, т. е. на веб-странице через них будет «просвечивать» фон; в современном варианте формата GIF можно хранить анимированные изображения; используется сжатие без потерь, т. е. при сжатии изображение не искажается;
- **PNG** (англ. *Portable Network Graphics* — переносимые сетевые изображения; файлы с расширением png) — формат, поддерживающий как режим истинного цвета, так и кодирование с палитрой; части изображения могут быть прозрачными и даже полупрозрачными (32-битное кодирование RGBA, где четвёртый байт задает прозрачность); изображение сжимается без искажения; анимация не поддерживается.

Свойства рассмотренных форматов сведены в таблицу 2.10.

Таблица 2.10

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

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

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

Векторное кодирование

Для чертежей, схем, карт применяется другой способ кодирования, который позволяет не терять качество при изменении размеров изображения. Рисунок строится из простейших геометрических фигур (**графических примитивов**): линий, многоугольников, сглаженных кривых, окружностей, эллипсов. Такой рисунок называется векторным.



Векторный рисунок — это рисунок, построенный из простейших геометрических фигур, параметры которых (размеры, координаты вершин, углы наклона, цвет контура и заливки) хранятся в виде чисел.

Векторный рисунок можно «разобрать» на части, растащив мышью его элементы, а потом снова собрать полное изображение (рис. 2.21).

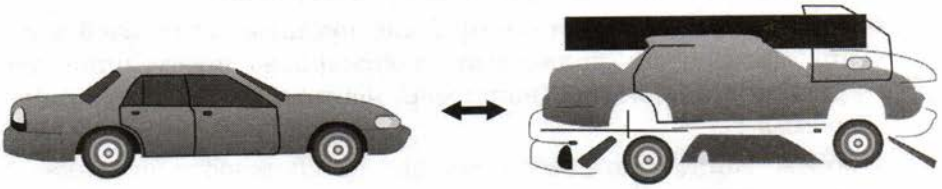


Рис. 2.21

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

При векторном кодировании для отрезка хранятся координаты его концов, для прямоугольников и ломаных — координаты вершин. Окружность и эллипс можно задать координатами прямоугольника, в который вписана фигура. Сложнее обстоит дело со сглаженными кривыми. На рисунке 2.22 изображена линия с опорными точками *А*, *Б*, *В*, *Г* и *Д*.

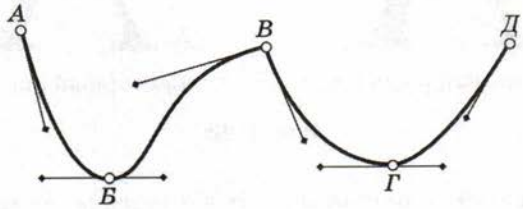


Рис. 2.22

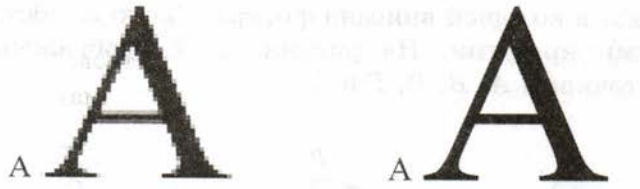
У каждой из этих точек есть «рычаги» (*управляющие линии*), перемещая концы этих рычагов, можно регулировать наклон касательной и кривизну всех участков кривой. Если оба рычага находятся на одной прямой, получается сглаженный узел (*Б* и *Г*), если нет, то угловой узел (*В*). Таким образом, форма этой кривой полностью задаётся координатами опорных точек и координатами рычагов. Кривые, заданные таким образом, называют *кривыми Безье* в честь их изобретателя — французского инженера Пьера Безье.

Векторный рисунок можно рассматривать как программу, в соответствии с которой строится изображение на конкретном

устройстве вывода, с учётом особенностей этого устройства (например, разрешения экрана).

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

- при кодировании нет потери информации, если изображение может быть полностью разложено на простейшие геометрические фигуры (например, чертеж, схема, карта, диаграмма);
- объём файлов напрямую зависит от сложности рисунка — чем меньше элементов, тем меньше места занимает файл; как правило, векторные рисунки значительно меньше по объёму, чем растровые;
- при изменении размера векторного рисунка не происходит никакого искажения формы элементов, при увеличении наклонных линий не появляются «ступеньки», как при растровом кодировании (рис. 2.23).



Растровый рисунок Векторный рисунок

Рис. 2.23

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

Среди форматов векторных рисунков отметим следующие:

- **WMF** (англ. *Windows Metafile* — метафайл Windows; файлы с расширением wmf и emf) — стандартный формат векторных рисунков в операционной системе Windows;
- **CDR** (файлы с расширением cdr) — формат векторных рисунков программы CorelDRAW;
- **AI** (файлы с расширением ai) — формат векторных рисунков программы Adobe Illustrator;
- **SVG** (англ. *Scalable Vector Graphics* — масштабируемые векторные изображения; файлы с расширением svg) — векторная графика для веб-страниц в Интернете.

Вопросы и задания

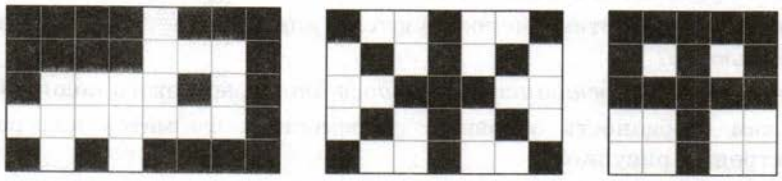
1. Какие два принципа кодирования рисунков используются в компьютерной технике?
2. Почему не удается придумать единый метод кодирования рисунков, пригодный во всех ситуациях?
3. В чём состоит идея растрового кодирования? Что такое растр?
4. Что такое пиксель?
5. Что такое дискретизация? Почему она необходима?
6. Что теряется при дискретизации? Почему?
7. Как уменьшить потерю информации при дискретизации? Что при этом ухудшается?
8. Что такое разрешение? В каких единицах оно измеряется?
9. Что такое режим истинного цвета (True Color)?
10. Что такое кодирование с палитрой? В чем его принципиальное отличие от режима истинного цвета?
11. Какие устройства используются для ввода изображений в компьютер?
12. В чём состоят *достоинства* и *недостатки* растрового кодирования?
13. В чём особенность основных современных форматов кодирования растровых рисунков?
14. Какие форматы поддерживают рисунки с прозрачными и полупрозрачными областями?
15. В каких форматах целесообразно сохранять фотографии? Рисунки с чёткими границами?
16. Как можно уменьшить объём файла растрового формата, в котором хранится рисунок? Чем при этом придётся пожертвовать?
17. Как компьютер определяет, что находится в файле — текст, рисунок, звук или видео?
18. Что такое кривые Безье?
19. Почему при увеличении растрового рисунка появляются «ступеньки»?
20. Что такое векторное кодирование? В чём его отличие от растрового? Каковы преимущества и недостатки растрового и векторного кодирования?
21. В каких задачах используют векторное кодирование?
22. Какие форматы векторных рисунков вы знаете? Приведите примеры.

Подготовьте сообщение:

а) «Цветовая модель Lab»
б) «Цветовая модель HSB»
в) «Цветовые профили устройств»
г) «Преобразования между цветовыми моделями»
д) «Кривые Безье»
е) «Формат BMP»
ж) «Формат GIF»
з) «Формат JPEG»
и) «Формат SVG»

Задачи

1. Постройте двоичные коды для чёрно-белых рисунков и запишите их в шестнадцатеричной системе счисления:



Какие сложности у вас возникли? Как их можно преодолеть?

2. Постройте чёрно-белый рисунок шириной 8 пикселей, закодированный шестнадцатеричной последовательностью $2466FF6624_{16}$.
3. Постройте чёрно-белый рисунок шириной 5 пикселей, закодированный шестнадцатеричной последовательностью $3A53F88_{16}$.
4. Рисунок размером 10 × 15 см кодируется с разрешением 300 ppi. Оцените количество пикселей в этом рисунке.
5. Постройте шестнадцатеричный код для цветов, имеющих RGB-коды (100, 200, 200), (30, 50, 200), (60, 180, 20), (220, 150, 30).
6. Как бы вы назвали цвета, заданные на веб-странице в виде кодов #CCCCCC, #FFCCCC, #CCCCFF, #000066, #FF66FF, #CCFFFF, #992299, #999900, #99FF99? Найдите десятичные значения составляющих RGB-кода.
7. Что такое глубина цвета? Как связаны глубина цвета и объём файла?
8. Какова глубина цвета, если в рисунке используется 65 536 цветов? 256 цветов? 16 цветов?

9. Для жёлтого цвета найдите красную, зелёную и синюю составляющие при 12-битном кодировании.
10. Сколько места в файле занимает палитра, в которой используются 64 цвета? 128 цветов?
11. Сколько байтов будет занимать код рисунка размером 40 × 50 пикселей в режиме истинного цвета? При кодировании с палитрой 256 цветов? При кодировании с палитрой 16 цветов? В чёрно-белом варианте (два цвета)?
12. Сколько байтов будет занимать код рисунка размером 80 × 100 пикселей в кодировании с глубиной цвета 12 битов на пиксель?
13. Для хранения растрового изображения размером 32 × 32 пикселя отвели 512 байтов памяти. Каково максимально возможное число цветов в палитре изображения?
14. Для хранения растрового изображения размером 128 × 128 пикселей отвели 4 килобайта памяти. Каково максимально возможное число цветов в палитре изображения?
15. В процессе преобразования растрового графического файла количество цветов уменьшилось с 1024 до 32. Во сколько раз уменьшился информационный объём файла?
16. В процессе преобразования растрового графического файла количество цветов уменьшилось с 512 до 8. Во сколько раз уменьшился информационный объём файла?
17. Разрешение экрана монитора — 1024 × 768 точек, глубина цвета — 16 битов. Какой объём памяти требуется для хранения полноэкранного изображения в данном графическом режиме?
18. После преобразования растрового 256-цветного графического файла в чёрно-белый формат (2 цвета) его размер уменьшился на 70 байтов. Каков был размер исходного файла (без учёта заголовка)?
19. Сколько памяти нужно для хранения 64-цветного растрового графического изображения размером 32 × 128 точек?
20. Какова ширина (в пикселях) прямоугольного 64-цветного растрового изображения, информационный объём которого 1,5 Мбайт, если его высота вдвое меньше ширины?
21. Какова ширина (в пикселях) прямоугольного 16-цветного растрового изображения, информационный объём которого 1 Мбайт, если его высота вдвое больше ширины?

§ 17
Кодирование звуковой и видеоинформации

Оцифровка звука

Звук — это колебания среды (воздуха, воды), которые воспринимает человеческое ухо. С помощью микрофона звук преобразуется в **аналоговый** электрический **сигнал** (см. § 7). В любой момент времени аналоговый сигнал на выходе микрофона (ток или напряжение) может принимать любое значение в некотором интервале (рис. 2.24).

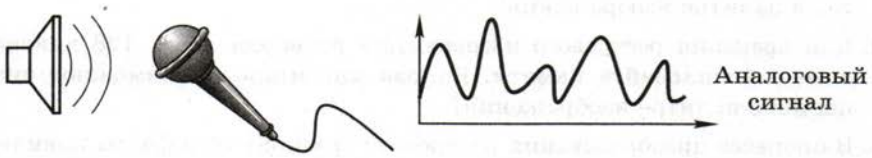


Рис. 2.24

Как вы знаете, современные компьютеры обрабатывают только дискретные сигналы (двоичные коды). Поэтому для работы со звуком необходима *звуковая карта*[1] — специальное устройство, которое преобразует аналоговый сигнал, полученный с микрофона, в двоичный код, т. е. в цепочку нулей и единиц. Эта процедура называется оцифровкой.

Оцифровка — это преобразование аналогового сигнала в цифровой код.

Ситуация напоминает ту, с которой мы столкнулись при кодировании рисунка: любая линия состоит из бесконечного числа точек, поэтому, чтобы закодировать «по точкам», нужна бесконечная память. Здесь тоже придётся использовать **дискретизацию** — представить аналоговый сигнал в виде набора чисел, т. е. записать в память только значения сигнала в отдельных точках, взятых с некоторым шагом T по времени (рис. 2.25).

[1] В современных персональных компьютерах функции звуковой карты часто выполняет специальная микросхема материнской платы — аппаратный аудиокодек.

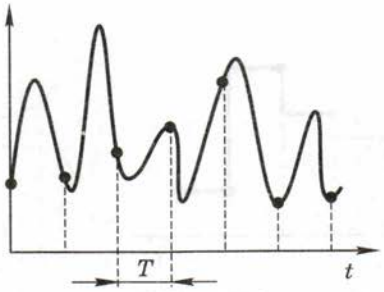


Рис. 2.25

Число T называется **интервалом дискретизации**, а обратная ему величина $1/T$ — **частотой дискретизации**. Частота дискретизации обозначается буквой f и измеряется в герцах (Гц) и килогерцах (кГц). Один герц — это один отсчёт в секунду, а 1 кГц — 1000 отсчётов в секунду. Чем больше частота дискретизации, тем точнее мы записываем сигнал, тем меньше информации теряем. Однако при этом возрастает количество отсчётов, т. е. информационный объём закодированного звука.

Для кодирования звука в компьютерах чаще всего используются частоты дискретизации 8 кГц (минимальное качество, достаточное для распознавания речи), 11 кГц, 22 кГц, 44,1 кГц (звуковые компакт-диски), 48 кГц (фильмы в формате DVD), а также 96 кГц и 192 кГц (высококачественный звук в формате DVD-audio). Выбранная частота влияет на качество цифрового звука. Дело в том, что наушники и звуковые колонки — это аналоговые (не цифровые) устройства, и при проигрывании звука через звуковую карту компьютеру нужно как-то восстановить исходный аналоговый сигнал и передать его на наушники или звуковые колонки. В памяти есть только значения, снятые с интервалом T, остальная информация была потеряна при кодировании. В простейшем случае по ним можно восстановить ступенчатый сигнал, который будет существенно отличаться от исходного (до кодирования) (рис. 2.26). В современных звуковых картах для повышения качества звука этот ступенчатый сигнал сглаживается с помощью специальных фильтров.

Для повышения качества звука, т. е. для большего соответствия между сигналом, принятым микрофоном, и сигналом, выведенным из компьютера на колонки, нужно увеличивать частоту дискретизации, однако при этом, как вы уже знаете, увеличивается и объём файла. Как же выбрать оптимальную частоту при

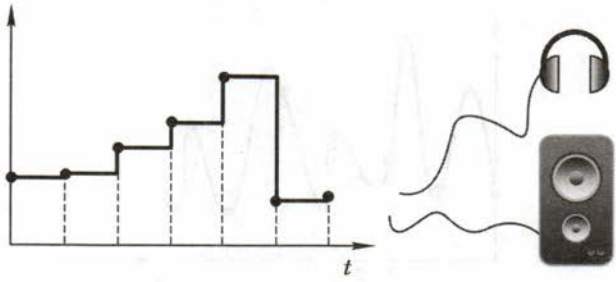


Рис. 2.26

кодировании? Ответ на этот вопрос во многом зависит от свойств звука, который нужно закодировать.

С точки зрения математики, любой сигнал можно представить в виде суммы очень большого числа колебаний разных частот (гармоник). Если выбрать частоту дискретизации больше, чем удвоенная частота самой быстрой гармоники, то теоретически по отдельным отсчётам можно точно восстановить исходный аналоговый сигнал. Этот результат известен в радиотехнике как *теорема Котельникова–Шеннона*.

К сожалению, на практике всё несколько сложнее. Дело в том, что в реальных сигналах содержатся гармоники с очень высокими частотами, так что частота дискретизации, полученная с помощью теоремы Котельникова–Шеннона, будет также высока и объём файла — недопустимо велик. Однако средний человек слышит только звуки с частотами от 16 Гц до 20 кГц, поэтому все частоты выше 20 кГц можно «потерять» практически без ухудшения качества звука (человек не почувствует разницу!). Удвоив эту частоту (по теореме Котельникова–Шеннона), получаем оптимальную частоту дискретизации около 40 кГц, которая обеспечивает наилучшее качество, различимое на слух. Поэтому при высококачественном цифровом кодировании звука на компакт-дисках и в видеофильмах чаще всего используют частоты 44,1 и 48 кГц. Более низкие частоты применяют тогда, когда важно всячески уменьшать объём звуковых данных (например, для трансляции радиопередач через Интернет), даже ценой ухудшения качества.

Кроме того что при кодировании звука выполняется дискретизация с потерей информации, нужно учитывать, что на хранение одного отсчёта в памяти отводится ограниченное место. При этом вносятся дополнительные ошибки.

Представим себе, что на один отсчёт выделяется 3 бита. При этом код каждого отсчёта — это целое число от 0 до 7. Весь диапазон возможных значений сигнала, от 0 до максимально допустимого, делится на 8 полос, каждой из которых присваивается номер (код). Все отсчёты, попавшие в одну полосу, получают одинаковый код (рис. 2.27).

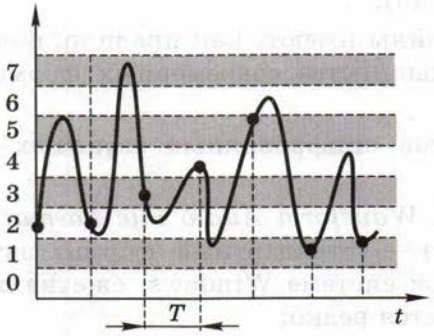


Рис. 2.27

Преобразование измеренного значения сигнала в целое число называется **дискретизацией по уровню** или **квантованием**. Эту операцию выполняет аналого-цифровой преобразователь (АЦП) — специальный блок звуковой карты.

Разрядность кодирования — это число битов, используемое для хранения одного отсчёта.

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

Объём данных, полученный после оцифровки звука, зависит от разрядности кодирования и частоты дискретизации. Например, если используется 16-разрядное кодирование с частотой 44 кГц, то за 1 с выполняется 44 000 измерений сигнала, и каждое из измеренных значений занимает 16 битов (2 байта). Поэтому за 1 секунду накапливается $44\ 000 \cdot 2 = 88\ 000$ байтов данных, а за 1 минуту: $88\ 000 \cdot 60 = 5\ 280\ 000$ байтов ≈ 5 Мбайт.

Если записывается стереозвук (левый и правый каналы), это число нужно удвоить.

С помощью оцифровки можно закодировать любой звук, который принимает микрофон. В частности, это единственный способ кодирования человеческого голоса и различных природных звуков (шума прибоя, шелеста листвы и т. п.).

Однако у этого метода есть и ***недостатки***:

- при оцифровке звука всегда есть потеря информации (из-за дискретизации);
- звуковые файлы имеют, как правило, большой размер, поэтому в большинстве современных форматов используется сжатие.

Среди **форматов оцифрованных звуковых файлов** наиболее известны:

- **WAV** (англ. *Waveform Audio File Format*; файлы с расширением wav) — стандартный формат звуковых файлов в операционной системе Windows; сжатие данных возможно, но используется редко;
- **MP3** (файлы с расширением mp3) — самый популярный формат звуковых файлов, использующий сжатие с потерями: для значительного уменьшения объёма файла снижается качество кодирования для тех частот, которые практически неразличимы для человеческого слуха;
- **WMA** (англ. *Windows Media Audio*; файлы с расширением wma) — формат звуковых файлов, разработанный фирмой Microsoft; чаще всего используется сжатие для уменьшения объема файла;
- **Ogg Vorbis** (файлы с расширением ogg) — свободный (не требующий коммерческих лицензий) формат сжатия звука с потерями.

Все эти форматы являются **потоковыми**, т. е. можно начинать прослушивание до того момента, как весь файл будет получен (например, из Интернета).

Инструментальное кодирование звука

Существует еще один, принципиально иной способ кодирования звука, который можно применить только для кодирования инструментальных мелодий. Он основан на **стандарте MIDI** (англ. *Musical Instrument Digital Interface* — цифровой интерфейс музыкальных инструментов). В отличие от оцифрованного звука в таком формате хранятся последовательность нот, коды инстру-

ментов (можно использовать 128 мелодических и 47 ударных инструментов), громкость, тембр, время затухания каждой ноты и т. д. Фактически это программа, предназначенная для проигрывания звуковой картой, в памяти которой хранятся образцы звуков реальных инструментов (*волновые таблицы*, англ. *wave tables*).

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

Звук, закодированный с помощью стандарта MIDI, хранится в файлах с расширением mid. Существуют специальные клавиатуры, которые позволяют вводить звук и сразу сохранять его в формате mid.

Для проигрывания MIDI-файла используют *синтезаторы* — электронные устройства, имитирующие звук реальных инструментов (рис. 2.28). Простейшим синтезатором является звуковая карта компьютера.

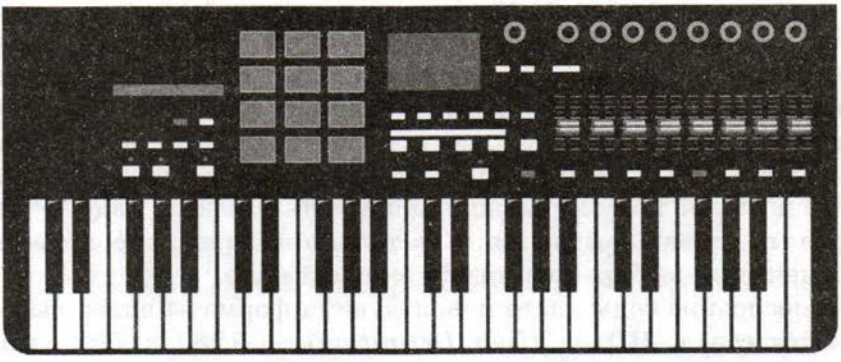


Рис. 2.28

Главные *достоинства* инструментального кодирования:

- кодирование мелодии (нотной записи) происходит без потери информации;
- файл имеет значительно меньший объём по сравнению с оцифрованным звуком той же длительности.

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

Кодирование видеоинформации

Для того чтобы сохранить видео в памяти компьютера, нужно закодировать звук и изменяющееся изображение, причём требуется обеспечить их *синхронность* (одновременность). Для кодирования звука чаще всего используют оцифровку с частотой 48 кГц. Изображение состоит из отдельных растровых рисунков, которые меняются с частотой не менее 25 кадров в секунду, так что глаз человека воспринимает смену кадров как непрерывное движение. Это значит, что для каждой секунды видео нужно хранить в памяти 25 изображений.

Если используется размер 768 × 576 точек (стандарты PAL/SECAM) и глубина цвета 24 бита на пиксель, то закодированная 1 секунда видео (без звука) будет занимать примерно 32 Мбайт, а 1 минута — около 1,85 Гбайт. Это недопустимо много, поэтому в большинстве форматов видеоизображений используется сжатие с потерями. Это значит, что некоторые незначительные детали теряются, но «обычный» человек (непрофессионал) не почувствует существенного ухудшения качества. Основная идея такого сжатия заключается в том, что за короткое время изображение изменяется очень мало, поэтому можно запомнить базовый кадр, а затем сохранять только изменения. Как только изображение существенно изменится, выбирается новый базовый кадр.

В последние годы часто используются форматы видео высокой чёткости (англ. **HD** — *High Definition*) — 1280 × 720 точек и 1920 × 1080 точек, предназначенные для просмотра на широкоформатных экранах с соотношением сторон 16:9.

Наиболее известны следующие **видеоформаты**:

- **AVI** (англ. *Audio Video Interleave* — чередующиеся звук и видео; файлы с расширением avi) — формат видеофайлов, разработанных фирмой Microsoft для системы Windows; может использовать разные алгоритмы сжатия;

- **WMV** (англ. *Windows Media Video*; файлы с расширением wmv) — система кодирования видео, разработанная фирмой Microsoft; может использовать разные алгоритмы сжатия;
- **MPEG** (файлы с расширением mpg, mpeg) — формат кодирования видеоинформации, использующий один из лучших алгоритмов сжатия, который разработала экспертная группа по вопросам движущегося изображения (англ. *Motion Picture Experts Group*);
- **MP4** (файлы с расширением mp4) — формат видеофайлов, позволяющий хранить несколько потоков видео высокой чёткости, а также субтитры;
- **MOV** (англ. *Quick Time Movie*; файлы с расширением mov) — формат видеофайлов, разработанный фирмой Apple;
- **WebM** — открытый (не требующий оплаты лицензии) видеоформат, который поддерживается в современных браузерах без установки дополнительных модулей.

Вопросы и задания

1. Что такое аналоговый сигнал?
2. Какие вы знаете аналоговые приборы?
3. Почему аналоговые компьютеры были вытеснены цифровыми?
4. Что такое оцифровка? Если ли потеря информации при оцифровке? Почему?
5. Что такое интервал дискретизации и частота дискретизации?
6. Как связаны частота дискретизации с потерей информации и объёмом файла?
7. Какие частоты дискретизации сейчас используются?
8. От чего зависит выбор частоты дискретизации?
9. Почему частоты дискретизации более 48 кГц применяются очень редко?
10. Как происходит вывод закодированного звукового сигнала на колонки или наушники?
11. Что такое дискретизация по уровню?

12. Какое устройство выполняет дискретизацию при записи звука?
13. Что такое разрядность кодирования звука? На что она влияет?
14. В чём *достоинства* и *недостатки* оцифровки?
15. Какие форматы файлов для хранения оцифрованного звука вы знаете?
16. Что такое потоковый звук?
17. Что такое инструментальное кодирование?
18. Что такое волновая таблица?
19. Что такое многоканальный звук?
20. В файлах с каким расширением хранится звук, закодированный с помощью стандарта MIDI?
21. Что такое синтезатор?
22. В чём *достоинства* и *недостатки* инструментального кодирования звука?
23. Почему MIDI-файлы могут звучать по-разному на разной аппаратуре?
24. Что такое синхронность?
25. Какая частота дискретизации звука чаще всего используется при кодировании видеофильмов?
26. Почему при кодировании видео используется частота не менее 25 кадров в секунду?
27. Почему компьютерные фильмы чаще всего хранятся в сжатом виде?
28. Что означает сжатие с потерями? В чём состоит его основная идея при кодировании видео?
29. Какие форматы видео вы знаете?

Подготовьте сообщение

а) «Как устроена звуковая карта?»
б) «Стандарт MIDI»
в) «Что такое кодек?»
г) «Что такое медиаконтейнер?»
д) «Формат MP3»
е) «Свободные звуковые и видеоформаты»

Задачи

1. Заполните пустые ячейки таблицы, вычислив объёмы звуковых файлов (без сжатия):

Частота дискретизации, кГц	8	8	11	11	22	22	44,1	44,1	48	48
Глубина кодирования, битов	8	8	16	16	16	8	24	8	8	24
Моно/стерео	моно	стерео	моно	стерео	моно	стерео	моно	стерео	стерео	стерео
Время звучания, с	16	8	64	32	32	32	256	128	4	4
Объём файла, Кбайт										

2. Заполните пустые ячейки таблицы, вычислив время звучания записи (объёмы файлов приведены без учёта сжатия):

Частота дискретизации, кГц	8	8	11	11	22	22	44,1	44,1	48	48
Глубина кодирования, битов	16	24	8	8	8	24	16	24	16	16
Моно/стерео	моно	стерео	моно	стерео	моно	стерео	моно	стерео	моно	стерео
Объём файла, Кбайт	125	375	1375	1375	6875	4125	11025	33075	375	1875
Время звучания, с										

3. Заполните пустые ячейки таблицы, вычислив глубину кодирования звука (объёмы файлов приведены без учёта сжатия):

Частота дискретизации, кГц	8	8	11	11	22	22	44,1	44,1	48	48
Моно/стерео	моно	стерео	моно	стерео	моно	стерео	моно	стерео	моно	моно
Время звучания, с	16	4	238	64	320	16	256	64	40	80
Объём файла, Кбайт	375	125	4125	4125	20625	1375	11025	11025	1875	11250
Глубина кодирования, битов										

4. Заполните пустые ячейки таблицы, вычислив частоту дискретизации звука (объёмы файлов приведены без учёта сжатия):

Глубина кодирования, битов	16	8	16	24	16	16	8	24	16	24
Моно/стерео	моно	стерео	стерео	стерео	стерео	моно	моно	стерео	моно	стерео
Время звучания, с	64	4	64	64	16	128	320	8	4	4
Объём файла, Кбайт	1375	375	11025	4125	1375	11025	6875	375	375	1125
Частота дискретизации, кГц										

5. Кадры видеозаписи закодированы в режиме истинного цвета (24 бита на пиксель) и сменяются с частотой 25 кадров в секунду, запись содержит стереофонический звук. Остальные параметры для разных вариантов заданы в таблице. Оцените объём 1 минуты видеозаписи в мегабайтах (с точностью до десятых). Сколько минут такой записи поместится на стандартный CD-диск объёмом 700 Мбайт?

Ширина кадра, пиксели	320	320	640	640	720	720	720	720	1920	1920
Высота кадра, пиксели	240	240	480	480	480	480	576	576	1080	1080
Частота дискретизации, кГц	11	48	48	48	22	48	22	48	48	48
Глубина кодирования звука, битов	24	16	24	16	16	16	24	24	16	24
Степень сжатия	10	8	6	4	10	12	8	6	8	10
Объём файла, Мбайт										
Поместится на CD-диск, минут										

Практические работы к главе 2

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

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

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

Самое важное в главе 2

- Кодирование — это преобразование информации в форму, удобную для её хранения, передачи и обработки. Компьютеры обрабатывают информацию в двоичном коде, в котором используется два знака (обычно 0 и 1).
- Данные, с которыми работает компьютер, — дискретные. Чтобы можно было автоматически обрабатывать аналоговую информацию (например, рисунки и звуки), её нужно дискретизировать — перевести в дискретный вид. Это, как правило, связано с потерей части информации.
- Система счисления — это способ записи чисел. Для компьютерной техники наиболее важны двоичная система счисления (в ней представляются данные в компьютере) и шестнадцатеричная система (такая форма записи компьютерных данных легче воспринимается человеком).
- Все типы данных в компьютерах кодируются в виде чисел, представленных в двоичной системе счисления.

Глава 3
Логические основы компьютеров

§ 18
Логика и компьютер

В быту мы часто используем слова «логика», «логично». Логика (от древнегреческого λογος — «мысль, рассуждение») — это наука о том, как правильно рассуждать, делать выводы, доказывать утверждения.

Логика — это наука о законах и формах правильного мышления.

В естественном языке рассуждения связаны с самыми разными предметами и понятиями, и поэтому исследовать всё это многообразие достаточно сложно. Древнегреческий философ **Аристотель** стал основоположником **формальной логики**, которая отвлекается от конкретного содержания понятий и изучает общие правила построения правильных выводов из известной информации, которая считается истинной. Формальная логика изучает логические высказывания.



Аристотель
(384–322 гг. до н. э.)

Логическое высказывание — это повествовательное предложение, про которое можно однозначно сказать, истинно оно или ложно.

Используя это определение, проверим, можно ли считать логическими высказываниями следующие предложения:

1. Сейчас идёт дождь.
2. Вчера жирафы улетели на север.
3. Красиво!

4. Который час?
5. В городе *N* живут более 2 миллионов человек.
6. Посмотрите на улицу.
7. У квадрата 10 сторон, и все разные.
8. История — интересный предмет.

Здесь высказываниями являются только предложения 1, 2 и 7, остальные не удовлетворяют определению. Утверждения 3 и 4 — это не повествовательные предложения. Предложение 5 станет высказыванием только в том случае, если *N* заменить на название конкретного города, так как о неизвестном городе нельзя ничего сказать. Предложение 6 — это призыв к действию, а не утверждение. Утверждение 8 кто-то считает истинным, а кто-то ложным — нет однозначности. Его можно более строго сформулировать в виде «По мнению *N*, история — интересный предмет». Для того чтобы оно стало высказыванием, нужно заменить *N* на имя человека.

Какая же связь между логикой и компьютерами? В классической формальной логике высказывание может быть истинно или ложно, третий вариант исключается[1]. Если обозначить истинное значение единицей, а ложное — нулём, то получится, что формальная логика представляет собой правила выполнения операций с нулями и единицами, т. е. с двоичными кодами. Как вы помните, именно такой способ используется в компьютерах для кодирования всех видов информации. Поэтому оказалось, что обработку данных можно свести к выполнению логических операций. Важный шаг в этом направлении сделал английский математик **Джордж Буль**. Он предложил применить для исследования логических высказываний математические методы. Позже этот раздел математики получил название алгебра логики или алгебра высказываний.



Дж. Буль (1815–1864)

[1] Существуют неклассические логические системы, например трёхзначная логика, где кроме «истинно» и «ложно» есть ещё состояние «не определено» (или «возможно»).

Алгебра логики — это математический аппарат, с помощью которого записывают, вычисляют, упрощают и преобразуют логические высказывания.

Алгебра логики определяет правила выполнения операций с логическими величинами, которые могут быть обозначены как 0 («ложь») и 1 («истина»), т. е. с двоичными данными. Используя эти правила, можно строить запоминающие элементы в компьютере и выполнять арифметические действия. О том, как это сделать, вы узнаете в этой главе.

Вопросы и задания

1. Объясните значения слов «логика», «формальная логика», «алгебра логики».
2. Чем отличается формальная логика от «обычной», «бытовой»?
3. Что такое высказывание?
4. Можно ли считать высказываниями следующие предложения?
   а) Не плачь, девчонка!
   б) Почему я водовоз?
   в) Купите слоника!
   г) Клубника очень вкусная.
   д) Сумма X и Y равна 36.

Подготовьте сообщение

«Информатика и логика»

§ 19
Логические операции

Высказывания бывают простые и сложные. Простые высказывания нельзя разделить на более мелкие высказывания. Примеры простых высказываний: «Сейчас идёт дождь»; «Форточка открыта». Сложные (составные) высказывания строятся из простых с помощью **логических связок** (операций) «И», «ИЛИ», «НЕ», «если..., то», «тогда и только тогда».

В алгебре логики высказывания обычно обозначаются латинскими буквами. Таким образом, мы уходим от конкретного содер-

жания высказываний, нас интересует только их истинность или ложность. Например, можно обозначить буквой A высказывание «Сейчас идет дождь», а буквой B — высказывание «Форточка открыта». Так как высказывания могут быть истинными или ложными, введённые символы A и B можно рассматривать как логические переменные, которые могут принимать два возможных значения: «ложь» (0) и «истина» (1). Из простых высказываний строятся сложные высказывания:

НЕ A: «Сейчас нет дождя».
НЕ B: «Форточка закрыта».
A И B: «Сейчас идёт дождь и открыта форточка».
A ИЛИ B: «Сейчас идёт дождь или открыта форточка».
ЕСЛИ A, ТО B: «Если сейчас идёт дождь, то форточка открыта».

Кроме этих высказываний есть ещё и другие, которые можно получить из двух исходных. С некоторыми из них мы также познакомимся.

Операции «НЕ», «И» и «ИЛИ» используются чаще других. Оказывается, с их помощью можно выразить любую логическую операцию, поэтому эти три операции считают основными, **базовыми**.

Операция «НЕ»

Операцию «НЕ» называют **отрицанием** или **инверсией** (англ. *inverse* — обратный). В алгебре логики всего два возможных значения (0 и 1), поэтому логические отрицание — это переход от одного значения к другому, от 1 к 0 или наоборот. Если высказывание A истинно, то НЕ A ложно, и наоборот.

Операция «НЕ» может обозначаться по-разному. Выражение НЕ A в алгебре логики записывается как $\overline{A}$ или $\neg A$, в языках программирования Паскаль и Бейсик — как `not A`, в языке Си — как `!A`.

Операцию «НЕ» можно задать в виде таблицы (рис. 3.1).

A	$\overline{A}$
0	1
1	0

Рис. 3.1

Эта таблица состоит из двух частей: слева перечисляются все возможные значения исходной величины (их всего два — 0 и 1), а в последнем столбце записывают результат выполнения логической операции для каждого из этих вариантов. Такая таблица называется **таблицей истинности** логической операции.

Таблица истинности задаёт **логическую функцию**, т. е. правила преобразования входных логических значений в выходные.

Операция «И»

Пусть есть два высказывания: A — «Сейчас идёт дождь», B — «Форточка открыта». Сложное высказывание «A И B» выглядит так: «Сейчас идёт дождь и форточка открыта». Оно будет истинным в том и только в том случае, когда оба высказывания A и B истинны одновременно.

Для понимания операции «И» можно представить себе простую схему, в которой для включения лампочки используются два выключателя, соединённых последовательно (рис. 3.2). Чтобы лампочка загорелась, нужно обязательно включить оба выключателя. Вместе с тем, чтобы выключить лампочку, достаточно выключить любой из них.

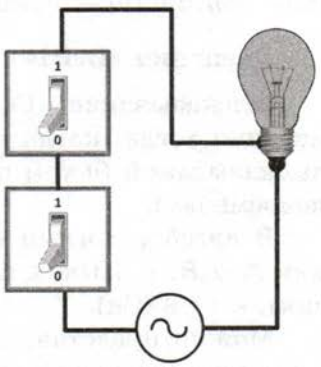


Рис. 3.2

Операция «И» (в отличие от «НЕ») выполняется с двумя логическими значениями, которые мы обозначим как A и B.

Результат этой операции в алгебре логики записывают как $A \cdot B$, $A \wedge B$ или A & B. В языках программирования используют обозначения, такие как `A and B` (Паскаль, Бейсик), `A && B` (Си).

В таблице истинности (рис. 3.3) будет уже не один столбец с исходными данными, а два. Число строк также выросло с 2 до 4, поскольку для 2 битов (A и B) мы получаем 4 разных комбинации значений: 00, 01, 10 и 11. Эти строки расположены в определённом порядке: двоичные числа, полученные соединением значений A и B, идут в порядке возрастания (слева от таблицы на рис. 3.3 они переведены в десятичную систему). Как следует из определения операции, в последнем столбце будет всего одна единица, для варианта $A = B = 1$.

	A	B	$A \cdot B$
0	0	0	0
1	0	1	0
2	1	0	0
3	1	1	1

Рис. 3.3

Легко проверить, что этот результат можно получить «обычным» умножением A на B, поэтому операцию «И» называют **логическим умножением**. Кроме того, с точки зрения обычной математики, эта операция выбирает минимальное из исходных значений. Существует ещё одно название операции «И» — **конъюнкция** (лат. *conjunctio* — союз, связь).

Операция «ИЛИ»

Высказывание «Сейчас идёт дождь или форточка открыта» истинно тогда, когда истинно хотя бы одно из входящих в него высказываний (в том числе когда истинны оба высказывания одновременно).

В алгебре логики операция «ИЛИ» обозначается как $A + B$ или $A \vee B$, в языках программирования — `A or B` (Паскаль, Бейсик), `A || B` (Си).

Можно представить себе схему с двумя выключателями, соединёнными параллельно (рис. 3.4). Чтобы лампочка загорелась, достаточно включить хотя бы один из выключателей. Чтобы выключить лампочку, необходимо обязательно выключить оба выключателя. В последнем столбце таблице истинности (рис. 3.5) будет только один ноль, для варианта $A = B = 0$.

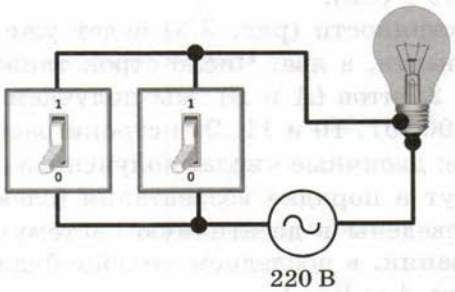


Рис. 3.4

A	B	$A+B$
0	0	0
0	1	1
1	0	1
1	1	1

Рис. 3.5

Операцию «ИЛИ» называют **логическим сложением**, потому что она похожа на обычное математическое сложение. Единственное отличие — в последней строке таблицы истинности: в математике 1 + 1 равно 2, а в алгебре логики — 1. Можно считать, что в результате применения операции «ИЛИ» из исходных значений выбирается наибольшее. Другое название этой операции — **дизъюнкция** (лат. *disjunctio* — разделение).

В учебнике для обозначения операций «И» и «ИЛИ» мы будем использовать знаки умножения и сложения ($A \cdot B$ и $A + B$). Это очень удобно потому, что они привычны для нас и позволяют легко увидеть аналогию с обычной математикой.

Доказано, что операций «НЕ», «И» и «ИЛИ» достаточно для того, чтобы записать с их помощью любую логическую операцию, которую только можно придумать. Далее мы рассмотрим ещё три логические операции и покажем, как их можно представить через операции «НЕ», «И» и «ИЛИ».

Операция «исключающее ИЛИ»

Операция «исключающее ИЛИ» отличается от обычного «ИЛИ» только тем, что её результат равен 0, если оба значения равны 1 (последняя строка в таблице истинности). То есть результат этой операции — истина в том и только в том случае, когда два значения не равны (рис. 3.6).

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

Рис. 3.6

Операции «исключающее ИЛИ» соответствует русская пословица «Либо пан, либо пропал», где выполнение обоих условий одновременно невозможно. Операция «исключающее ИЛИ» в алгебре логики обозначается знаком $\oplus$, в языке Паскаль — `xor` (`A xor B`), а в языке Си — знаком ^ (`A ^ B`). Эту операцию можно представить через базовые операции (НЕ, И, ИЛИ) следующим образом:

$$A \oplus B = \overline{A} \cdot B + A \cdot \overline{B}.$$

Пока мы не можем вывести это равенство, но можем доказать его (или опровергнуть, т. е. доказать, что оно неверное). Для этого достаточно для всех возможных комбинаций A и B вычислить значения выражения, стоящего в правой части равенства, и сравнить их со значениями выражения $A \oplus B$ для тех же исходных данных. Поскольку провести такие вычисления в уме достаточно сложно, сначала вычислим значения $\overline{A}$, $\overline{B}$, $\overline{A} \cdot B$ и $A \cdot \overline{B}$, а потом уже $\overline{A} \cdot B + A \cdot \overline{B}$. В таблице истинности появятся дополнительные столбцы для промежуточных результатов (рис. 3.7).

A	B	$\overline{A}$	$\overline{B}$	$\overline{A} \cdot B$	$A \cdot \overline{B}$	$\overline{A} \cdot B + A \cdot \overline{B}$	$A \oplus B$
0	0	1	1	0	0	0	0
0	1	1	0	1	0	1	1
1	0	0	1	0	1	1	1
1	1	0	0	0	0	0	0

Рис. 3.7

Легко видеть, что выражение $\overline{A} \cdot B + A \cdot \overline{B}$ совпадает с $A \oplus B$ для всех комбинаций значений. Это значит, что равенство доказано.

Операция «исключающее ИЛИ» иначе называется **разделительной дизъюнкцией** (это значит «один или другой, но не оба вместе») или **сложением по модулю два**. Второе название связано с тем, что её результат равен остатку от деления арифметической суммы $A + B$ на 2:

$$A \oplus B = (A + B) \bmod 2.$$

Здесь mod обозначает операцию взятия остатка от деления. Из таблицы истинности видно, что $A \oplus B = 1$ тогда и только тогда, когда $A \neq B$.

Операция «исключающее ИЛИ» обладает интересными свойствами. По таблице истинности несложно проверить, что

$$A \oplus 0 = A, \quad A \oplus 1 = \overline{A}, \quad A \oplus A = 0.$$

Для доказательства этих равенств можно просто подставить в них $A = 0$ и $A = 1$. Теперь докажем, что

$$(A \oplus B) \oplus B = A. \qquad (1)$$

Подставляя в левую часть $B = 0$, получим $(A \oplus 0) \oplus 0 = A \oplus 0 = A$. Аналогично для $B = 1$ имеем $(A \oplus 1) \oplus 1 = \overline{A} \oplus 1 = A$. Это означает, что формула (1) справедлива для любых значений B. Отсюда следует важный вывод: если два раза применить операцию «исключающее ИЛИ» с одним и тем же значением переменной, мы восстановим исходное значение. В этом смысле «исключающее ИЛИ» — обратимая операция (кроме неё обратима также операция «НЕ» — если применить её дважды, мы вернёмся к исходному значению).

Формулу (1) можно использовать для шифрования данных. Пусть A и B — двоичные коды одинаковой длины. Чтобы зашифровать данные (A), используя ключ B, надо применить операцию «исключающее ИЛИ» отдельно для каждого разряда A и B. Для расшифровки ещё раз применяется «исключающее ИЛИ» с тем же ключом B. Нужно отметить, что такой метод шифрования очень нестойкий: для достаточно длинных текстов его легко «взломать» частотным анализом.

Импликация

Мы часто используем логическую связку «если..., то», например: «Если пойдёт дождь, то я надену плащ» или «Если все стороны прямоугольника равны, то это квадрат». В логике эта связка называется импликацией[1] (следованием) и обозначается стрелкой: $A \to B$ («если A, то B», «из A следует B»). Таблица истинности операции импликации показана на рис. 3.8.

A	B	$A \to B$
0	0	1
0	1	1
1	0	0
1	1	1

Рис. 3.8

[1] От лат. *implicatio* — сплетение, тесная связь.

Разобраться с импликацией будет легче, если мы рассмотрим конкретное высказывание, например, такое: «Если хорошо работаешь, то получаешь большую зарплату». Обозначим буквами два простых высказывания: A — «Вы хорошо работаете» и B — «Вы получаете большую зарплату». Понятно, что если высказывание $A \to B$ истинно, то все, кто хорошо работают ($A = 1$), должны получать большую зарплату ($B = 1$). Если же кто-то работает хорошо ($A = 1$), а получает мало ($B = 0$), то высказывание $A \to B$ ложно.

Лодыри и бездельники ($A = 0$) могут получать как маленькую ($B = 0$), так и большую зарплату ($B = 1$), это не нарушает истинность высказывания $A \to B$. Иногда, определяя импликацию, говорят так: из истины следует истина, а из лжи — что угодно. Это значит, что при ложном высказывании A высказывание B может быть как ложно, так и истинно.

Нужно обратить внимание на разницу между высказываниями вида «если A, то B» в обычной жизни и в алгебре логики. В быту мы чаще всего имеем в виду, что существует причинно-следственная связь между A и B, т. е. именно A вызывает B. Алгебра логики не устанавливает взаимосвязь явлений; истинность высказывания $A \to B$ говорит только о возможности такой связи. Например, с точки зрения алгебры логики может быть истинным высказывание «Если Вася — студент, то Петя — лыжник».

Импликация чаще всего используется при решении логических задач, так как формулировку вида «если A, то B» можно записать как $A \to B = 1$.

Для импликации (в отличие от других изученных ранее операций с двумя переменными) не действует переместительный закон: если в записи $A \to B$ поменять местами A и B, то результат изменится: $A \to B \neq B \to A$. Внешне это видно по стрелке, которая указывает «направление».

Импликацию можно заменить на выражение, использующее только базовые операции (здесь — только «НЕ» и «ИЛИ»):

$$A \to B = \overline{A} + B.$$

Доказать это равенство вы уже можете самостоятельно.

Эквивалентность

Эквивалентность (также называется **эквиваленцией**) — это логическая операция, которая соответствует связке «тогда и только тогда». Высказывание $A \leftrightarrow B$ истинно в том и только в том случае, когда $A = B$ (см. таблицу истинности — рис. 3.9).

A	B	$A \leftrightarrow B$
0	0	1
0	1	0
1	0	0
1	1	1

Рис. 3.9

Возможно, вы заметили, что эквивалентность — это обратная операция для операции «исключающее ИЛИ» (проверьте по таблицам истинности), т. е.

$$A \leftrightarrow B = \overline{A \oplus B}.$$

Здесь черта сверху, охватывающая всё выражение в правой части равенства, означает отрицание (инверсию), которое применяется к результату вычисления выражения $A \oplus B$, а не к отдельным высказываниям.

Можно заменить эквивалентность выражением, которое включает только базовые логические операции:

$$A \leftrightarrow B = \overline{A} \cdot \overline{B} + A \cdot B.$$

Это равенство вы можете доказать (или опровергнуть) самостоятельно.

Другие логические операции

Таблицы истинности операций с двумя переменными содержат 4 строки и отличаются только значением последнего столбца. Поэтому любая новая комбинация нулей и единиц в этом столбце даёт новую логическую операцию (логическую функцию). Всего их, очевидно, столько, сколько существует четырёхразрядных двоичных чисел, т. е. $16 = 2^4$. Из тех операций, которые мы ещё не рассматривали, наиболее интересны две — **штрих Шеффера** («И-НЕ», англ. **nand** — «not and», рис. 3.10):

$$A \mid B = \overline{A \cdot B}$$

и **стрелка Пирса** («ИЛИ-НЕ», англ. **nor** — «not or», рис. 3.11):

$$A \downarrow B = \overline{A + B}.$$

Штрих Шеффера

A	B	$A \mid B$
0	0	1
0	1	1
1	0	1
1	1	0

Рис. 3.10

Стрелка Пирса

A	B	$A \downarrow B$
0	0	1
0	1	0
1	0	0
1	1	0

Рис. 3.11

Особенность этих операций состоит в том, что с помощью любой одной из них можно записать произвольную логическую операцию. Например, операции «НЕ», «И» и «ИЛИ» (базовый набор) выражаются через штрих Шеффера так:

$$\overline{A} = A \mid A, \quad A \cdot B = \overline{A \mid B} = (A \mid B) \mid (A \mid B),$$
$$A + B = \overline{A} \mid \overline{B} = (A \mid B) \mid (B \mid B).$$

Эти равенства можно доказать через таблицы истинности.

Логические выражения

Обозначив простые высказывания буквами (переменными) и используя логические операции, можно записать любое высказывание в виде логического выражения. Например, пусть система сигнализации должна дать аварийный сигнал, если вышли из строя два из трёх двигателей самолета. Обозначим высказывания:

A — «Первый двигатель вышел из строя».
B — «Второй двигатель вышел из строя».
C — «Третий двигатель вышел из строя».
X — «Аварийная ситуация».

Тогда логическое высказывание X можно записать в виде логического выражения (**логической формулы**):

$$X = (A \cdot B) + (A \cdot C) + (B \cdot C). \qquad (2)$$

Здесь мы выполнили формализацию.

Формализация — это переход от конкретного содержания к формальной записи с помощью некоторого языка.

В логических выражениях операции выполняются в следующем порядке:

1) действия в скобках;
2) отрицание (НЕ);
3) логическое умножение (И);
4) логическое сложение (ИЛИ) и операция «исключающее ИЛИ»;
5) импликация;
6) эквивалентность.

Такой порядок означает, что все скобки в выражении (2) для X можно убрать. Порядок вычисления выражения можно, так же, как и для арифметических выражений, определить с помощью дерева (рис. 3.12). Вычисление начинается с листьев, корень — это самая последняя операция.

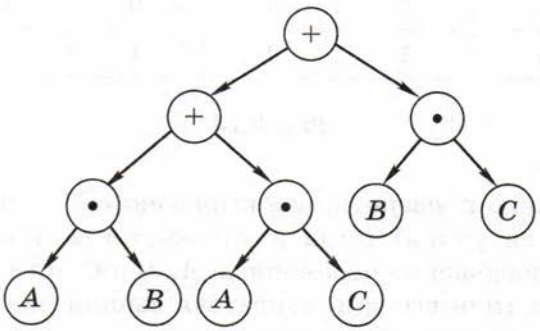


Рис. 3.12

Здесь каждая операция выполняется с двумя значениями. Такие операции называются **бинарными** (лат. *bis* — дважды) или **двуместными**.

Операции, которые выполняются над одной величиной, называют **унарными** (лат. *uno* — один) или **одноместными**. Пример унарной логической операции — отрицание (операция «НЕ»).

Любую формулу можно задать с помощью таблицы истинности, которая показывает, чему равно значение логического выра-

жения при всех возможных комбинациях значений исходных переменных. Сложные выражения удобно разбить на несколько более простых, сначала вычислить значения этих промежуточных величин, а затем — окончательный результат.

Рассмотрим формулу (2). Выражение в правой части зависит от трёх переменных, поэтому существует $2^3 = 8$ комбинаций их значений. Таблица истинности выглядит так, как показано на рис. 3.13.

A	B	C	$A \cdot B$	$A \cdot C$	$B \cdot C$	X
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	0	1	1
1	0	0	0	0	0	0
1	0	1	0	1	0	1
1	1	0	1	0	0	1
1	1	1	1	1	1	1

Рис. 3.13

По ней можно проверить, что выражение X истинно (возникает аварийная ситуация) тогда и только тогда, когда любые две (или все три) логические переменные A, B и C истинны (равны 1), т. е. любые два (или все три) двигателя вышли из строя. Поэтому формализация задачи выполнена верно.

Из таблицы 3.13 видно, что при некоторых значениях переменных значение X истинно, а при некоторых — ложно. Такие выражения называют **вычислимыми**.

Высказывание «Вася — школьник, или он не учится в школе» всегда истинно (для любого Васи). Оно может быть записано в виде логического выражения $A + \overline{A}$. Выражение, истинное при любых значениях переменных, называется **тождественно** истинным или **тавтологией**.

Высказывание «Сегодня безветрие, и дует сильный ветер» никогда не может быть истинным. Соответствующее логическое

выражение $A \cdot \overline{A}$ всегда ложно, оно называется **тождественно ложным** или **противоречием**.

Если два выражения принимают одинаковые значения при всех значениях переменных, они называются **равносильными** или **тождественно равными**. Например, выражения $A \to B$ и $\overline{A} + B$ равносильны, равенство $A \to B = \overline{A} + B$ называют **тождеством**. Равносильные выражения определяют одну и ту же логическую функцию, т. е. при одинаковых исходных данных приводят к одинаковым результатам.

Некоторые задачи

Рассмотрим ряд задач, в которых требуется исследовать логическое выражение.

Задача 1. Каково наибольшее целое число X, при котором истинно следующее высказывание?

$$A = (90 < X^2) \to (80 > (X+2)^2)$$

Сначала удобно заменить импликацию по формуле $A \to B = \overline{A} + B$. Отрицание для высказывания $90 < X^2$ запишется как $90 \geq X^2$, поэтому

$$A = (90 \geq X^2) \text{ или } (80 > (X+2)^2).$$

В этой задаче нас интересуют только целые числа. Поэтому условие $90 \geq X^2$ можно заменить на ($|X| \leq 9$ или $-9 \leq X \leq 9$), а условие $80 > (X+2)^2$ — на ($|X+2| \leq 8$ или $-10 \leq X \leq 6$). Таким образом, требуется выбрать наибольшее целое число, которое входит в один или в другой промежуток (рис. 3.14).

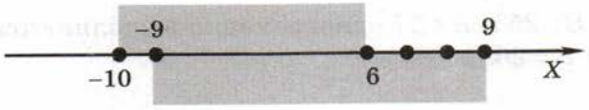


Рис. 3.14

Это число 9.

Задача 2. A, B и C — целые числа, для которых истинно высказывание

$$X = \overline{(A = B)} \cdot ((A > B) \to (B > C)) \cdot ((B > A) \to (C > B)).$$

Чему равно B, если $A = 27$ и $C = 25$?

Данное сложное высказывание состоит из трёх простых:

$$\overline{(A = B)},\quad (A > B) \to (B > C),\quad (B > A) \to (C > B).$$

Они связаны операцией «И», т. е. должны быть истинными одновременно. Из условия $(A = B) = 1$ сразу следует, что $A \neq B$. Далее можно использовать несколько способов решения.

Способ 1 (решение «по частям»). Предположим, что $A > B$, тогда из второго условия получаем $1 \to (B > C)$. Это выражение может быть истинно тогда и только тогда, когда $(B > C) = 1$, поэтому имеем $A > B > C$. Этому условию соответствует только число 26. Одно решение найдено.

Теперь проверим вариант $A < B$. Из второго условия получаем $0 \to (B > C)$, это выражение истинно при любом B. Проверяем третье условие: получаем $1 \to (C > B) = 1$; это выражение может быть истинно тогда и только тогда, когда $C > B$, и тут мы получили противоречие, потому что нет такого числа B, для которого $C > B > A$. Таким образом, правильный ответ — 26, других решений нет.

Способ 2. Раскрываем импликацию через операции «ИЛИ» и «НЕ»:

$$(A > B) \to (B > C) = \overline{(A > B)} + (B > C) = (B \geq A) + (B > C),$$
$$(B > A) \to (C > B) = \overline{(B > A)} + (C > B) = (B \leq A) + (B < C).$$

Учитывая, что раньше мы выяснили, что $A \neq B$, можно заменить знаки $\geq$ и $\leq$ соответственно на $>$ и $<$. Подставляя значения $A = 27$ и $C = 25$, упрощаем выражения:

$$(B > 27) + (B > 25) = (B > 25),$$
$$(B < 27) + (B < 25) = (B < 27).$$

Условия $B > 25$ и $B < 27$ одновременно выполняются только для целого числа $B = 26$.

Вопросы и задания

1. Даны два высказывания: A — «В Африке водятся жирафы» и B — «В Мурманске идёт снег». Постройте из них различные сложные высказывания.
2. Дано высказывание «Винни-Пух любит мёд, и дверь в дом открыта». Как бы вы сформулировали отрицание этого высказывания?
3. Что такое таблица истинности?

4. Почему таблица истинности для операции «НЕ» содержит две строки, а таблицы для других изученных операций — четыре? Сколько строк в таблице истинности выражения с тремя переменными? С четырьмя? С пятью?
5. В каком порядке обычно записываются значения переменных в таблице истинности? Зачем это нужно?
6. Когда истинно высказывание *A* И *B*? *A* ИЛИ *B*?
7. Какие электрические схемы можно использовать для иллюстрации операций «И» и «ИЛИ»?
8. Какие знаки применяют для обозначения операций «НЕ», «И», «ИЛИ»?
9. Почему операция «И» называется логическим умножением, а «ИЛИ» — логическим сложением?
10. В чём различие арифметического и логического сложения?
11. Сколько можно определить различных логических функций с двумя переменными? С тремя переменными?
12. Чем отличается операция «исключающее ИЛИ» от операции «ИЛИ»?
13. Почему операция «исключающее ИЛИ» называется сложением по модулю 2?
14. Как записать выражение $A \oplus B$ с помощью базового набора операций (НЕ, И, ИЛИ)?
15. Как можно доказать или опровергнуть логическое равенство?
16. Какими интересными свойствами обладает операция «исключающее ИЛИ»?
17. Что значит выражение «обратимая операция»? Какие изученные логические операции являются обратимыми?
18. Какое свойство операции «исключающее ИЛИ» позволяет использовать ее для простейшего шифрования?
19. Чем отличается смысл высказывания «если *A*, то *B*» в обычной речи и в математической логике?
20. Запишите в виде логической формулы высказывание «Если утюг горячий, то лоб холодный».
21. Запишите в виде логической формулы высказывание «Неверно, что если утюг горячий, то лоб холодный». Можно ли в этом случае сразу сказать, каким является утюг и каким — лоб?
22. Как выразить импликацию через операции «НЕ» и «ИЛИ»? Докажите полученное тождество.
23. Как выразить эквивалентность через операции «НЕ», «И» и «ИЛИ»? Докажите полученное тождество.
24. Чем интересны операции «штрих Шеффера» и «стрелка Пирса»?
25. Докажите тождества, позволяющие представить базовые логические операции через штрих Шеффера. Попробуйте построить и доказать аналогичные тождества для операции «стрелка Пирса».
26. Что такое формализация?

27. В каком порядке выполняются действия в логических выражениях?
28. Что можно сделать для того, чтобы изменить естественный порядок действий?
29. Какие операции называются бинарными и унарными? Приведите примеры унарных и бинарных операций в математике.
30. Поясните разницу между терминами «логическое выражение» и «логическая функция».
31. Можно ли сказать, что таблица истинности однозначно определяет:
    а) логическое выражение;
    б) логическую функцию?
32. Что такое вычислимое логическое выражение?
33. Что такое тавтология? Противоречие? Приведите примеры.
34. Что такое равносильные выражения?



Подготовьте сообщение

а) «Логическая операция "Штрих Шеффера"»
б) «Логическая операция "Стрелка Пирса"»
в) «Шифрование с помощью операции "исключающее ИЛИ"»



Задачи

1. Составьте деревья для вычисления логических выражений и таблицы истинности этих выражений:

а) $\overline{A \cdot B} + A \cdot B$;
б) $A \cdot B + \overline{A} \cdot \overline{B} + A \cdot \overline{B}$;
в) $(A + B) \cdot (\overline{A} + \overline{B}) \cdot (A + \overline{B})$;
г) $A \cdot \overline{B} + B \cdot \overline{C} + C \cdot \overline{A}$;
д) $A \cdot \overline{B} \cdot C + \overline{A} \cdot B \cdot \overline{C} + B \cdot C$;
е) $A \cdot (\overline{B} \cdot C + \overline{A}) \cdot (\overline{C} + B)$;
ж) $\overline{A \cdot C} + \overline{B \cdot \overline{C}}$;
з) $\overline{(A + C)} + \overline{(B + \overline{C})}$;
и) $\overline{\overline{(\overline{A} \cdot C)} \cdot \overline{(\overline{B} \cdot C)}}$;
к) $A \cdot (C + B \cdot \overline{C}) + C \cdot \overline{(A + B)}$;
л) $A \cdot (C + \overline{(\overline{B} + C)}) + B \cdot \overline{(A \cdot C)}$.

2. Составьте деревья для вычисления логических выражений и таблицы истинности этих выражений:

а) $(A \to B) + (\overline{A} \to \overline{B})$;
б) $(\overline{A} \to B) \cdot (A \to \overline{B})$;
в) $(A \cdot B) \to (\overline{A} + \overline{B})$;
г) $(A + \overline{B}) \to (A \cdot B)$;
д) $(A \to \overline{B}) \cdot (A + C) \cdot (\overline{A} \to C)$;
е) $(\overline{A} \to \overline{B}) \to (A \to C)$;
ж) $A \cdot B \to (B + \overline{C})$;
з) $(\overline{A} \to B) \to \overline{(\overline{A} \to \overline{C})}$;
и) $(A \leftrightarrow B) + (\overline{A} \leftrightarrow B)$;
к) $(A \leftrightarrow \overline{B}) + (A \leftrightarrow C) + (\overline{B} \leftrightarrow C)$.

3. Символом F обозначено одно из указанных ниже логических выражений от трёх аргументов: X, Y, Z. Дан фрагмент таблицы истинности выражения F. Какие из этих выражений могут соответствовать F?

X	Y	Z	F
1	1	1	1
1	1	0	1
1	0	0	1

а) $X + \overline{Y} + \overline{Z}$;
б) $X + Y + Z$;
в) $\overline{X} + Y + Z$;
г) $X + \overline{Y} + \overline{Z}$.

4. Для предыдущего задания определите, сколько различных логических функций соответствует заданной частичной таблице истинности.

5. Задано 5 строк таблицы истинности некоторого логического выражения с тремя переменными. Сколько различных логических функций ей соответствуют?

6. Символом F обозначено одно из указанных ниже логических выражений от трёх аргументов: X, Y, Z. Дан фрагмент таблицы истинности выражения F. Какие из этих выражений могут соответствовать F?

X	Y	Z	F
0	1	0	0
1	1	0	1
0	1	1	0

а) $\overline{X} + Y + \overline{Z}$;
б) $X \cdot Y \cdot \overline{Z}$;
в) $\overline{X} + \overline{Y} + Z$;
г) $X + \overline{Y} \cdot Z$.

7. Символом F обозначено одно из указанных ниже логических выражений от трёх аргументов: X, Y, Z. Дан фрагмент таблицы истинности выражения F. Какие из этих выражений могут соответствовать F?

X	Y	Z	F
1	0	0	1
0	0	0	1
1	1	1	0

а) $X \to (\overline{Y} + \overline{Z})$;
б) $\overline{X} \cdot \overline{Y} \cdot \overline{Z}$;
в) $\overline{X} + \overline{Y} + \overline{Z}$;
г) $X + Y + Z$.

8. Символом F обозначено одно из указанных ниже логических выражений от трех аргументов: X, Y, Z. Дан фрагмент таблицы истинности выражения F. Какие из этих выражений могут соответствовать F?

X	Y	Z	F
1	0	0	1
0	0	0	0
1	1	1	0

а) $X \cdot \overline{Y} \cdot \overline{Z}$;
б) $X \to (\overline{Y} + \overline{Z})$;
в) $X + Y + Z$;
г) $Y \to (X \cdot Z)$.

9. Символом F обозначено одно из указанных ниже логических выражений от трех аргументов: X, Y, Z. Дан фрагмент таблицы истинности выражения F. Какие из этих выражений могут соответствовать F?

X	Y	Z	F
0	0	0	0
0	1	1	1
1	0	0	1

а) $(X + \overline{Y}) \to Z$;
б) $(\overline{X} + Y) \to Z$;
в) $X + (Y \to Z)$;
г) $X + Y \cdot Z$.

10. Определите значение логического выражения $(X > 2) \to (X > 3)$ для $X = 1, 2, 3, 4$.

11. Определите значение логического выражения

$((X < 5) \to (X < 3)) \cdot ((X < 2) \to (X < 1))$ для $X = 1, 2, 3, 4$.

12. Определите значение логического выражения

$((X > 3) + (X < 3)) \to (X < 1)$ для $X = 1, 2, 3, 4$.

13. Определите значение логического выражения

$((X < 4) \to (X < 3)) \cdot ((X < 3) \to (X < 1))$ для $X = 1, 2, 3, 4$.

14. Определите значение логического выражения

$(X \cdot (X - 8) > 2 \cdot X - 25) \to (X > 7)$ для $X = 4, 5, 6, 7$.

15. Найдите все целые значения X, при которых логическое выражение $(X > 2) \to (X > 5)$ ложно.

16. Найдите все целые значения X, при которых логическое выражение $((X > 0) + (X > 4)) \to (X > 4)$ ложно.

17. Автопилот может работать, если исправен главный бортовой компьютер или два вспомогательных. Выполните формализацию и запишите логические формулы для высказываний «Автопилот работоспособен» и «Автопилот неработоспособен».

18. Каково наибольшее целое положительное число X, при котором истинно утверждение:

$$(X(X+3) > X^2+9) \to (X(X+2) \le X^2+11)?$$

19. Каково наибольшее целое положительное число X, при котором истинно утверждение:

$$(121 < X^2) \to (X > X+5)?$$

20. Каково наибольшее целое положительное число X, при котором ложно утверждение:

$$(X(X+6)+9 > 0) \to (X^2 > 45)?$$

21. Каково наибольшее целое положительное число X, при котором истинно утверждение:

$$(X^2-1 > 100) \to (X(X-1) < 100)?$$

22. Каково наибольшее целое положительное число X, при котором ложно утверждение:

$$(7X-3 < 75) \to (X(X-1) > 65)?$$

23. Известно, что для чисел A, B и C истинно утверждение

$$((C < A)+(C < B)) \cdot \overline{((C+1) < A) \cdot ((C+1) < B)}.$$

а) Чему равно C, если $A = 25$ и $B = 48$?
б) Чему равно C, если $A = 45$ и $B = 18$?

24. Известно, что для чисел A, B и C истинно утверждение

$$\overline{(A = B)} \cdot ((B < A) \to (2C > A)) \cdot ((A < B) \to (A > 2C)).$$

Чему равно A, если $C = 10$ и $B = 22$?

§ 20 Диаграммы Венна

Выражения, зависящие от небольшого количества переменных (обычно не более четырёх), удобно изображать в виде диаграмм, которые называют **диаграммами** Венна или **кругами Эйлера**.

На такой диаграмме каждой переменной соответствует круг, внутри которого она равна единице, а вне его — нулю. Круги могут пересекаться. Области, в которых рассматриваемое логическое выражение истинно, закрашиваются каким-либо цветом. На рисунке 3.14 приведены диаграммы для простейших операций с одной и двумя переменными. Серым цветом залиты области, где рассматриваемое выражение равно единице.

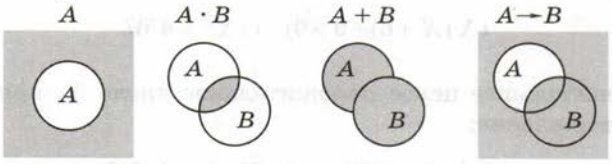
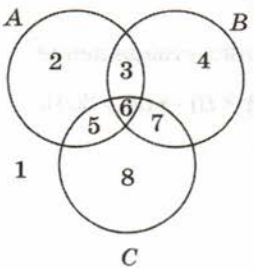


Рис. 3.14

Такие диаграммы часто используются при работе с множествами: операция «И» соответствует пересечению двух множеств, а «ИЛИ» — объединению.

Для трёх переменных диаграмма будет немного сложнее. Для каждой из областей показанной на рис. 3.15 диаграммы запишем логические выражения.



1: $\overline{A} \cdot \overline{B} \cdot \overline{C}$
2: $A \cdot \overline{B} \cdot \overline{C}$
3: $A \cdot B \cdot \overline{C}$
4: $\overline{A} \cdot B \cdot \overline{C}$
5: $A \cdot \overline{B} \cdot C$
6: $A \cdot B \cdot C$
7: $\overline{A} \cdot B \cdot C$
8: $\overline{A} \cdot \overline{B} \cdot C$

Рис. 3.15

Для того чтобы найти выражение для объединения двух или нескольких областей, надо применить логическое сложение (операцию «ИЛИ») к выражениям для всех составляющих. Например, выражение для объединения областей 3 и 4 имеет вид

$$3 + 4\text{: } A \cdot B \cdot \overline{C} + \overline{A} \cdot B \cdot \overline{C}.$$

Вместе с тем если не обращать внимания на область A, то можно заметить, что справедлива формула

$$3 + 4\text{: } B \cdot \overline{C}.$$

Это означает, что логические выражения в некоторых случаях можно упростить. Как это делается, вы узнаете в следующем параграфе.

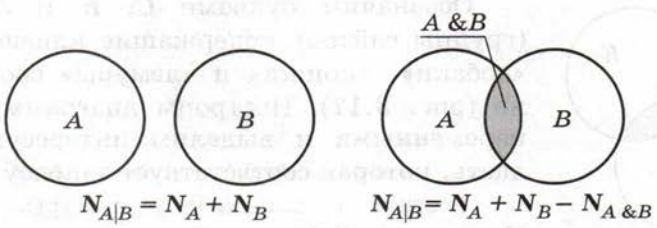
Диаграммы удобно применять для решения задач, в которых используются множества, например множества страниц, полученных от поисковой системы в ответ на какой-то запрос. Рассмотрим следующую задачу.

Задача 1. Известно количество страниц, которые находит поисковый сервер по следующим запросам (здесь символ «&» обозначает операцию «И», а «|» — операцию «ИЛИ»):

`собаки	кошки`	770
`кошки`	550
`собаки & кошки`	100

Сколько страниц будет выдано по запросу `собаки`?

Сначала попробуем рассмотреть задачу в общем виде и вывести формулу для её решения. Построим диаграмму с двумя областями A и B. Эти области могут быть разделены (рис. 3.16, *а*) или пересекаться (рис. 3.16, *б*).



$N_{A|B} = N_A + N_B$ $N_{A|B} = N_A + N_B - N_{A\,\&B}$

а *б*

Рис. 3.16

Обозначим через N_X число страниц, которые выдаются по запросу X. В первом случае, когда области не пересекаются, получаем очевидную формулу: $N_{A|B} = N_A + N_B$. Это значит, что количество страниц, полученных по запросу $A \mid B$, будет равно сумме результатов по отдельным запросам.

Во втором случае (рис. 3.16, *б*) сумма $N_A + N_B$ дважды включает общую область, т. е. результат запроса A & B. Поэтому формула изменяется:

$$N_{A|B} = N_A + N_B - N_{A\&B}.$$

Это более общий случай, справедливый и для рис. 3.16, *а*, где $N_{A\&B} = 0$. Для нашей задачи (область A — собаки, область B — кошки) получаем:

$$N_A = N_{A|B} - N_B + N_{A\&B} = 770 - 550 + 100 = 320.$$

Рассмотрим теперь более сложную задачу, с тремя областями.

Задача 2. Известно количество страниц, которые находит поисковый сервер по следующим запросам (здесь символ «&» обозначает операцию «И», а «|» — операцию «ИЛИ»):

собаки	200
кошки	250
лемуры	450
кошки | собаки	450
кошки & лемуры	40
собаки & лемуры	50

Сколько страниц найдёт этот сервер по запросу

(кошки | собаки) & лемуры?

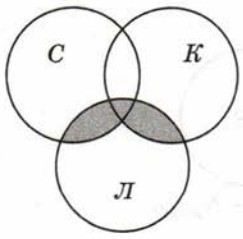


Рис. 3.17

Обозначим буквами *С*, *К* и *Л* области (группы сайтов), содержащие ключевые слова «собаки», «кошки» и «лемуры» соответственно (рис. 3.17). Построим диаграмму с тремя переменными и выделим интересующую область, которая соответствует запросу

(кошки | собаки) & лемуры.

На рисунке 3.17 эта область закрашена серым цветом.

В общем виде задача очень сложна. Попробуем найти какое-нибудь упрощающее условие. Например, выделим три условия:

собаки	200
кошки	250
кошки | собаки	450

Это означает, что область «кошки ИЛИ собаки» равна сумме областей «кошки» и «собаки», т. е. эти области не пересекаются! Таким образом, в нашем случае диаграмма выглядит, как показано на рис. 3.18.

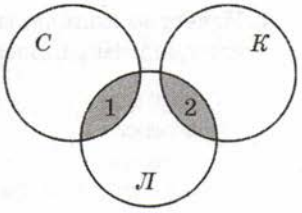


Рис. 3.18

Области 1 (собаки & лемуры) и 2 (кошки & лемуры) нам известны, они составляют соответственно 40 и 50 страниц, поэтому по запросу

(кошки | собаки) & лемуры

поисковый сервер выдаст 40 + 50 = 90 страниц.

Подготовьте сообщение

а) «Диаграммы Венна и теория множеств»
б) «Язык запросов поисковых систем»

Задачи

1. Используя диаграмму с тремя переменными (см. рис. 3.15), запишите логические выражения для объединения областей 2 + 5, 3 + 6, 4 + 7, 6 + 7, 5 + 6, 5 + 8, 7 + 8. Для каждой сложной области найдите два эквивалентных выражения.
2. Известно количество страниц, которые находит поисковый сервер по следующим запросам:

собаки	200
кошки	300
кошки | собаки	450

Сколько страниц найдет этот сервер по запросу

кошки & собаки?

3. Известно количество страниц, которые находит поисковый сервер по следующим запросам:

собаки	200
кошки	250
кошки & собаки	50

Сколько страниц найдёт этот сервер по запросу
кошки | собаки?

4. Известно количество страниц, которые находит поисковый сервер по следующим запросам:

собаки	200
кошки	250
лемуры	450
кошки | собаки	450
кошки & лемуры	40
собаки & лемуры	50

Сколько страниц найдёт этот сервер по запросу
кошки | собаки | лемуры?

5. Известно количество страниц, которые находит поисковый сервер по следующим запросам:

собаки	250
кошки	200
лемуры	500
собаки & лемуры	0
собаки & кошки	20
кошки & лемуры	10

Сколько страниц найдёт этот сервер по запросу
кошки | собаки | лемуры?

6. Известно количество страниц, которые находит поисковый сервер по следующим запросам:

собаки	120
кошки	270
лемуры	100
кошки | собаки	390
кошки & лемуры	20
собаки & лемуры	10

Сколько страниц найдёт этот сервер по запросу
кошки | собаки | лемуры?

***7.** Известно количество страниц, которые находит поисковый сервер по следующим запросам:

`собаки`	50
`кошки`	60
`лемуры`	70
`собаки | кошки`	80
`собаки | лемуры`	100
`лемуры & (собаки | кошки)`	20

Сколько страниц найдёт этот сервер по запросу

`кошки & (собаки | лемуры)`?

§ 21 Упрощение логических выражений

Законы алгебры логики

Для упрощения логических выражений используют законы алгебры логики. Они формулируются для базовых логических операций — «НЕ», «И» и «ИЛИ».

Закон	Для операции «И»	Для операции «ИЛИ»
двойного отрицания	$\overline{\overline{A}} = A$	
исключённого третьего	$A \cdot \overline{A} = 0$	$A + \overline{A} = 1$
операции с константами	$A \cdot 1 = A, \quad A \cdot 0 = 0$	$A + 1 = 1, \quad A + 0 = A$
повторения	$A \cdot A = A$	$A + A = A$
переместительный	$A \cdot B = B \cdot A$	$A + B = B + A$
сочетательный	$A \cdot (B \cdot C) = (A \cdot B) \cdot C$	$A + (B + C) = (A + B) + C$
распределительный	$A + B \cdot C = (A + B) \cdot (A + C)$	$A \cdot (B + C) = A \cdot B + A \cdot C$
поглощения	$A + A \cdot B = A$	$A \cdot (A + B) = A$
де Моргана	$\overline{A \cdot B} = \overline{A} + \overline{B}$	$\overline{A + B} = \overline{A} \cdot \overline{B}$

Закон двойного отрицания означает, что операция «НЕ» обратима: если применить ее два раза, логическое значение не изменится. Закон исключённого третьего основан на том, что в классической (двузначной) логике любое логическое выражение либо истинно, либо ложно («третьего не дано»). Поэтому если $A = 1$, то $\overline{A} = 0$ (и наоборот), так что произведение этих величин всегда равно нулю, а сумма — единице.

Операции с константами и закон повторения легко проверяются по таблицам истинности операций «И» и «ИЛИ». Переместительный и сочетательный законы выглядят вполне привычно, так же, как и в арифметике. Почти везде «работает» аналогия с алгеброй чисел, нужно только помнить, что в логике $1 + 1 = 1$, а не 2.

Распределительный закон для операции «ИЛИ» — это обычное раскрытие скобок. А вот для операции «И» мы видим незнакомое выражение, в алгебре чисел это равенство неверно. Доказательство можно начать с правой части, раскрыв скобки:

$$(A + B) \cdot (A + C) = A \cdot A + A \cdot C + B \cdot A + B \cdot C.$$

Дальше используем закон повторения ($A \cdot A = A$) и заметим, что

$$A + A \cdot C = A \cdot (1 + C) = A \cdot 1 = A.$$

Аналогично доказываем, что $A + B \cdot A = A \cdot (1 + B) = A$, таким образом,

$$(A + B) \cdot (A + C) = A + B \cdot C.$$



О. де Морган
(1806–1871)

Равенство доказано. Попутно мы доказали также и закон поглощения для операции «И» (для операции «ИЛИ» вы можете сделать это самостоятельно). Отметим, что из распределительного закона следует полезное тождество:

$$A + \overline{A} \cdot B = (A + \overline{A}) \cdot (A + B) = A + B.$$

Правила, позволяющие раскрывать отрицание сложных выражений, названы в честь шотландского математика и логика Огастеса (Августа) де Моргана. Обратите внимание, что при этом не просто «общее» отрицание переходит на отдельные выражения, но и операция «И» заменяет-

ся на «ИЛИ» (и наоборот). Доказать законы де Моргана можно с помощью таблиц истинности.

Теперь с помощью приведённых законов алгебры логики упростим полученное ранее логическое выражение для объединения областей 3 и 4 на диаграмме с тремя переменными (§ 20, рис. 3.15):

$$A \cdot B \cdot \overline{C} + \overline{A} \cdot B \cdot \overline{C} = (A + \overline{A}) \cdot B \cdot \overline{C} = B \cdot \overline{C}.$$

Здесь мы сначала вынесли общий множитель двух слагаемых за скобки, а затем применили закон исключённого третьего.

В общем случае можно рекомендовать такую последовательность действий.

1. Заменить все «небазовые» операции (исключающее ИЛИ, импликацию, эквивалентность и др.) на их выражения через базовые операции «НЕ», «И» и «ИЛИ».
2. Раскрыть отрицания сложных выражений по законам де Моргана так, чтобы операции отрицания остались только у отдельных переменных.
3. Используя вынесение общих множителей за скобки, раскрытие скобок и другие законы алгебры логики, упростить выражение.

Пример

$$(A + \overline{B}) \cdot \overline{(A + B)} \cdot (\overline{A} + C) = (A + \overline{B}) \cdot \overline{A} \cdot \overline{B} \cdot (\overline{A} + C) =$$
$$= (A \cdot \overline{A} + \overline{B} \cdot \overline{A}) \cdot \overline{B} \cdot (\overline{A} + C) = \overline{B} \cdot \overline{A} \cdot \overline{B} \cdot (\overline{A} + C) =$$
$$= \overline{A} \cdot \overline{B} \cdot \overline{B} \cdot (\overline{A} + C) = \overline{B} \cdot \overline{A} \cdot (\overline{A} + C) = \overline{B} \cdot \overline{A}.$$

Здесь последовательно использованы закон де Моргана, распределительный закон, закон исключённого третьего, переместительный закон, закон повторения, снова переместительный закон и закон поглощения.

Логические уравнения

Если приравнять два логических выражения, мы получим уравнение. Его решением будут значения переменных, при которых уравнение превращается в истинное равенство, т. е. когда значения левой и правой частей совпадают. Например, уравнение $A \cdot B = 1$ имеет единственное решение: $A = B = 1$, для остальных

комбинаций значений переменных левая часть равна нулю. В то же время уравнение $A+B=1$ имеет три решения: ($A=0$, $B=1$), ($A=1$, $B=0$) и $A=B=1$.

Пример 1. Требуется найти все решения уравнения

$$(\overline{(B+C)}\cdot A)\to(\overline{A}\cdot\overline{C}+D)=0.$$

Вспоминаем, что импликация равна нулю только тогда, когда первое выражение равно 1, а второе — 0. Поэтому исходное уравнение сразу разбивается на два:

$$\overline{(B+C)}\cdot A=1,\quad \overline{A}\cdot\overline{C}+D=0.$$

Первое уравнение с помощью закона де Моргана можно преобразовать к виду $\overline{B}\cdot\overline{C}\cdot A=1$, откуда сразу следует, что все три сомножителя должны быть равны 1. Это значит, что $A=1$, $B=0$ и $C=0$. Кроме того, из второго уравнения следует, что $D=0$. $A=1$ и $C=0$ удовлетворяют и второму уравнению тоже. Решение найдено, причём оно единственное.

Возможен *второй вариант* — упростить выражение. Заменяя импликацию по формуле $A\to B=\overline{A}+B$, получаем:

$$\overline{\overline{(B+C)}\cdot A}+\overline{A}\cdot\overline{C}+D=0.$$

Используем закон де Моргана:

$$B+C+\overline{A}+\overline{A}\cdot\overline{C}+D=0$$

и закон поглощения:

$$B+C+\overline{A}+D=0.$$

Для того чтобы логическая сумма была равна нулю, каждое слагаемое должно быть равно нулю, поэтому $A=1$, $B=C=D=0$.

Есть и *третий вариант* — построить таблицу истинности выражения в левой части и найти все варианты, при которых оно равно 0. Однако таблица истинности выражения с четырьмя переменными содержит $2^4=16$ строк, поэтому такой подход достаточно трудоёмок.

Пример 2. Требуется найти все решения уравнения

$$(A+\overline{B})\to(B\cdot C\cdot D)=1.$$

Преобразуем выражение, раскрыв импликацию через операции «НЕ» и «ИЛИ» и применив закон де Моргана:

$$\overline{A+\overline{B}}+B\cdot C\cdot D=\overline{A}\cdot B+B\cdot C\cdot D=1.$$

Если логическая сумма равна 1, то хотя бы одно слагаемое равно 1 (или оба одновременно).

Равенство $\overline{A} \cdot B = 1$ верно при $A = 0$, $B = 1$ и любых C и D. Поскольку есть всего 4 комбинации значений C и D, уравнение $\overline{A} \cdot B = 1$ имеет 4 решения:

A	B	C	D
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1

Второе уравнение, $B \cdot C \cdot D = 1$, даёт $B = C = D = 1$ при любом A, т. е. оно имеет два решения:

A	B	C	D
0	1	1	1
1	1	1	1

повторяется

Видим, что первое из этих решений уже было получено раньше, поэтому уравнение имеет всего пять разных решений. Заметим, что определить все повторяющиеся решения можно из уравнения $(\overline{A} \cdot B) \cdot (B \cdot C \cdot D) = 1$, которое имеет единственное решение $A = 0$, $B = C = D = 1$.

Пример 3. Требуется найти число решений уравнения

$$A \cdot B \cdot C + \overline{B} \cdot \overline{C} \cdot D = 0.$$

Здесь, в отличие от предыдущих задач, не нужно находить сами решения, интересует только их количество. Уравнение распадается на два:

$$A \cdot B \cdot C = 0 \text{ и } \overline{B} \cdot \overline{C} \cdot D = 0.$$

Каждое из них имеет достаточно много решений. Можно поступить следующим образом: сначала найти количество решений «обратного» уравнения, с единицей в правой части:

$$A \cdot B \cdot C + \overline{B} \cdot \overline{C} \cdot D = 1$$

и затем вычесть его из 16 (общего количества комбинаций четырёх переменных). Уравнение $A \cdot B \cdot C = 1$ имеет два решения: $A = B = C = 1$ и любое D (0 или 1). Второе уравнение, $\overline{B} \cdot \overline{C} \cdot D = 1$, тоже имеет два решения: A — любое, $B = C = 0$, $D = 1$. Среди этих четырёх решений нет повторяющихся, поэтому исходное уравнение имеет 16 – 4 = 12 решений.

Пример 4. Требуется найти число решений уравнения

$$(X_1 \to X_2) \cdot (X_2 \to X_3) \cdot (X_3 \to X_4) \cdot (X_4 \to X_5) \cdot (X_5 \to X_6) = 1.$$

Так как каждая логическая переменная может принимать только значения 0 и 1, можно представить решение в виде 6-битной цепочки. Например, цепочка 010110 означает, что $X_1 = X_3 = X_6 = 0$ и $X_2 = X_4 = X_5 = 1$.

Заметим, что импликация $A \to B$ ложна тогда и только тогда, когда $A = 1$ и $B = 0$. Поэтому в решении заданного уравнения не может встречаться последовательность 10, иначе какая-то импликация оказывается ложной и всё выражение будет равно 0. Поэтому существует всего 7 решений:

000000 000001 000011 000111 001111 011111 111111

Обратите внимание, что число решений логических уравнений, в отличие от «обычных уравнений», всегда конечно. Это связано с тем, что каждая переменная может принимать только два значения (0 и 1), и число разных комбинаций значений переменных конечно, оно равно 2^n, где n — это количество переменных. Поэтому уравнение с n переменными имеет не более 2^n решений.

Подготовьте сообщение

а) «Законы логики и правила алгебры: сходство и различия»
б) «Методы решения логических уравнений»
в) «Системы логических уравнений»

Задачи

1. Упростите логические выражения:

а) $A \cdot B \cdot \overline{A} \cdot B + B$;
б) $(A + B) \cdot (\overline{A} + \overline{B})$;
в) $A + A \cdot B + A \cdot C$;
г) $A + \overline{A} \cdot B + \overline{A} \cdot C$;
д) $A \cdot (A + B + C)$;
е) $A \cdot B + \overline{B} + \overline{A} \cdot B$;
ж) $(\overline{A} + B) \cdot \overline{C} \cdot (C + A \cdot \overline{B})$;
з) $\overline{A} \cdot \overline{C} + A \cdot B + \overline{A} \cdot C + A \cdot \overline{B}$;
и) $A \cdot (\overline{B} \cdot \overline{C} + B \cdot C) + A \cdot (B \cdot \overline{C} + \overline{B} \cdot C)$.

2. Упростите логические выражения:

а) $A \cdot \overline{(\overline{B}+C)}$;

б) $\overline{(A+\overline{B})}+\overline{(A+B)}+A \cdot B$;

в) $A+\overline{(A+B)}+\overline{A} \cdot B$;

г) $\overline{(A+\overline{B}+\overline{C})}$;

д) $\overline{(A+B)} \cdot A \cdot \overline{B}$;

е) $A+\overline{B \cdot \overline{C}}+\overline{(\overline{A}+B+\overline{C})}$;

ж) $(A+B+C) \cdot \overline{(\overline{A} \cdot \overline{B})}+C$;

з) $A \cdot \overline{(\overline{C}+\overline{B})}+\overline{(\overline{A}+B)} \cdot C+A \cdot C$;

и) $(A+B) \cdot (\overline{A}+B) \cdot (\overline{A}+\overline{B})$.

3. Упростите логические выражения:

а) $\overline{(A \to C)} \cdot C$;

б) $\overline{(\overline{A} \to \overline{B})}+\overline{(\overline{A} \to B)}+A \cdot B$;

в) $A+\overline{(\overline{A} \to B)}+\overline{(A+\overline{B})}$;

г) $\overline{(\overline{A} \to (B \to \overline{C}))}$;

д) $\overline{(\overline{A} \to B)} \cdot \overline{(A \to B)}$;

е) $A+\overline{B \cdot \overline{C}}+\overline{(A \to \overline{\overline{B} \cdot C})}$.

4. Решите уравнения:

а) $A+\overline{B}+(B \to (C+D))=0$;

б) $(A \to C)+B \cdot A+\overline{D}=0$;

в) $(\overline{A}+C) \to (\overline{B}+C+D)=0$;

г) $(A \to \overline{C})+\overline{B} \cdot C \cdot A+D=0$;

д) $(\overline{(B+C)} \cdot A) \to (\overline{(A+C)}+D)=0$;

е) $(A \to C) \cdot (A \to \overline{C}) \cdot (\overline{A} \to (C \cdot \overline{B} \cdot D))=1$.

5. Сколько различных решений имеют уравнения?

а) $A \cdot B+C \cdot D=1$;

б) $(A+B) \cdot (C+D)=1$;

в) $(A+B) \to (B \cdot C \cdot D)=0$;

г) $A \cdot \overline{B} \cdot C \cdot \overline{D} \cdot (E+\overline{E})=0$;

д) $(A+B+C) \cdot \overline{B} \cdot \overline{C} \cdot D=1$;

е) $(A \cdot B \cdot C) \to (\overline{C} \cdot D)=1$;

ж) $(A \to B) \cdot C+\overline{C} \cdot D=1$;

з) $(\overline{A}+\overline{B}+\overline{C}) \cdot (B+\overline{C}+\overline{D})=0$.

***6.** Сколько различных решений имеют уравнения?

а) $(A \to B) \cdot (B \to C) \cdot (C \to D)=0$;

б) $(A \to B) \cdot (B \to C) \cdot (C \to D) \cdot (D \to E) \cdot (D \to B)=1$;

в) $(A \to B) \cdot \overline{(B \to C)} \cdot (C \to D)=0$;

г) $(A \to B) \cdot \overline{(B \to C)} \cdot \overline{(C \to D)} \cdot (D \to E)=1$;

д) $A \to B \to C \to D \to E \to F=1$.

§ 22
Синтез логических выражений

До этого момента мы считали, что логическое выражение уже задано и нам надо что-то с ним сделать (построить таблицу истинности, упростить и т. п.). Такие задачи называются задачами **анализа** (от греческого αναλυσις — разложение), в них требуется исследовать заданное выражение. При проектировании различных логических устройств, в том числе и устройств компьютеров, приходится решать обратную задачу — строить логическое выражение по готовой таблице истинности, которая описывает нужное правило обработки данных. Эта задача называется задачей **синтеза** (от греческого συνθεσις — совмещение).

В качестве простейшего примера построим логическое выражение, тождественное операции импликации $X = A \to B$, по её таблице истинности (рис. 3.19).

A	B	X	
0	0	1	$\overline{A} \cdot \overline{B}$
0	1	1	$\overline{A} \cdot B$
1	0	0	
1	1	1	$A \cdot B$

Рис. 3.19

Способ 1. В таблице истинности мы выделяем все строки, где логическое выражение равно единице. Тогда искомое выражение может быть записано как логическая сумма выражений, каждое из которых истинно только в одном случае.

Например, выражение $\overline{A} \cdot \overline{B}$ истинно только при $A = 0$ и $B = 0$, т. е. только в первой строке таблицы. Выражение $\overline{A} \cdot B$ истинно только во второй строке, а $A \cdot B$ — только в последней.

Существует простое правило: *если в некоторой строке переменная равна нулю, она входит в произведение с отрицанием, а если равна 1, то без отрицания.*

Складывая выражения для всех отмеченных строк (кроме третьей, где функция равна нулю), получаем:

$$X = \overline{A} \cdot \overline{B} + \overline{A} \cdot B + A \cdot B.$$

Упрощаем это выражение:

$$X = \overline{A} \cdot (\overline{B} + B) + A \cdot B = \overline{A} + A \cdot B = (\overline{A} + A) \cdot (\overline{A} + B) = \overline{A} + B.$$

Таким образом, мы вывели формулу, которая позволяет заменить импликацию через операции «НЕ» и «ИЛИ».

Способ 2. Если в таблице истинности нулей меньше, чем единиц, удобнее сначала найти формулу для обратного выражения, $\overline{X}$, а потом применить операцию «НЕ». В данном случае выражение равно нулю в единственной строке, при $A = 1$ и $B = 0$, только в этой строке $\overline{X} = 1$, поэтому, используя предыдущий способ, получаем $\overline{X} = A \cdot \overline{B}$. Теперь остаётся применить операцию «НЕ» и закон де Моргана:

$$X = \overline{A \cdot \overline{B}} = \overline{A} + B.$$

Рассмотрим более сложный пример, когда выражение зависит от трёх переменных. В этом случае в таблице истинности будет 8 строк (рис. 3.20).

A	B	C	X	
0	0	0	1	$\overline{A} \cdot \overline{B} \cdot \overline{C}$
0	0	1	1	$\overline{A} \cdot \overline{B} \cdot C$
0	1	0	1	$\overline{A} \cdot B \cdot \overline{C}$
0	1	1	1	$\overline{A} \cdot B \cdot C$
1	0	0	0	
1	0	1	1	$A \cdot \overline{B} \cdot C$
1	1	0	0	
1	1	1	1	$A \cdot B \cdot C$

Рис. 3.20

Отметим все строки, где $X = 1$, и для каждой из них построим выражение, истинное только для этой комбинации переменных (см. рис. 3.20). Теперь выполним логическое сложение:

$$X = \overline{A} \cdot \overline{B} \cdot \overline{C} + \overline{A} \cdot \overline{B} \cdot C + \overline{A} \cdot B \cdot \overline{C} + \overline{A} \cdot B \cdot C + A \cdot \overline{B} \cdot C + A \cdot B \cdot C.$$

Упрощение этого выражения даёт:

$$X = \overline{A} \cdot \overline{B} \cdot (\overline{C} + C) + \overline{A} \cdot B \cdot (\overline{C} + C) + A \cdot C \cdot (\overline{B} + B) =$$
$$= \overline{A} \cdot \overline{B} + \overline{A} \cdot B + A \cdot C = \overline{A} \cdot (\overline{B} + B) + A \cdot C =$$
$$= \overline{A} + A \cdot C = (\overline{A} + A) \cdot (\overline{A} + C) = \overline{A} + C.$$

Используя второй способ, получаем:

$$\overline{X} = A \cdot \overline{B} \cdot \overline{C} + A \cdot B \cdot \overline{C} = A \cdot \overline{C} \cdot (\overline{B} + B) = A \cdot \overline{C}.$$

Тогда $X = \overline{A \cdot \overline{C}} = \overline{A} + C$. В данном случае второй способ оказался проще, потому что в столбце X таблицы истинности меньше нулей, чем единиц.

Способ 3. При небольшом количестве нулей можно использовать ещё один метод. Попробуем применить операцию «НЕ» к исходному выражению для $\overline{X}$, без предварительного упрощения:

$$X = \overline{A \cdot \overline{B} \cdot \overline{C} + A \cdot B \cdot \overline{C}}.$$

Применяя закон де Моргана, получим:

$$X = \overline{(A \cdot \overline{B} \cdot \overline{C})} \cdot \overline{(A \cdot B \cdot \overline{C})}.$$

Используя закон де Моргана для обеих скобок, находим:

$$X = (\overline{A} + B + C) \cdot (\overline{A} + \overline{B} + C).$$

Заметим, что выражение в каждой скобке *ложно* только для одной комбинации исходных данных, при которых $X = 0$.

Таким образом, третий способ заключается в том, чтобы для каждой строки в таблице истинности, где выражение равно 0, построить логическую сумму, в которую переменные, равные в этой строке единице, входят с инверсией, а равные нулю — без инверсии. Выражение для X — это произведение полученных сумм.

В нашем примере выражение упрощается с помощью распределительного закона для «И» и закона исключённого третьего:

$$X = (\overline{A} + B + C) \cdot (\overline{A} + \overline{B} + C) = (\overline{A} + C) + B \cdot \overline{B} = \overline{A} + C.$$

Неудивительно, что мы получили тот же ответ, что и раньше.

Иногда при упрощении выражений может потребоваться искусственный приём, который сначала вроде бы усложняет запись, но затем позволяет получить более простую форму. Например, рассмотрим выражение

$$X = \overline{A} \cdot B + \overline{A} \cdot C + \overline{B} \cdot C.$$

Учитывая, что $B+\overline{B}=1$, можно представить второе слагаемое в виде:

$$\overline{A}\cdot C=\overline{A}\cdot(B+\overline{B})\cdot C=\overline{A}\cdot B\cdot C+\overline{A}\cdot\overline{B}\cdot C.$$

Тогда получаем:

$$X=\overline{A}\cdot B+\overline{A}\cdot B\cdot C+\overline{A}\cdot\overline{B}\cdot C+\overline{B}\cdot C=\overline{A}\cdot B\cdot(1+C)+(\overline{A}+1)\cdot\overline{B}\cdot C=$$
$$=\overline{A}\cdot B+\overline{B}\cdot C.$$

Подготовьте сообщение

а) «Совершенные нормальные формы»
б) «Карты Карно»

Задачи

1. Постройте выражения для логических функций, заданных таблицами истинности. Используйте разные методы и сравните их.

а)

A	B	X
0	0	0
0	1	1
1	0	0
1	1	1

б)

A	B	X
0	0	1
0	1	0
1	0	1
1	1	1

в)

A	B	X
0	0	0
0	1	1
1	0	0
1	1	0

г)

A	B	X
0	0	0
0	1	0
1	0	1
1	1	0

2. Постройте выражения для логических функций, заданных таблицами истинности. Используйте разные методы и сравните их.

а)

A	B	C	X
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

б)

A	B	C	X
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

в)

A	B	C	X
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

§ 23
Предикаты и кванторы

В предыдущих параграфах мы видели, как алгебра логики позволяет нам записывать высказывания в виде формул и делать выводы. Однако с помощью алгебры логики невозможно доказать некоторые довольно простые утверждения. Рассмотрим такие высказывания:

«Все люди смертны».
«Сократ — человек».

Каждый из нас понимает, что если оба этих высказывания истинны, то Сократ тоже смертен. Однако алгебра высказываний не позволяет это доказать. В таких случаях приходится использовать другой математический аппарат, с которым мы познакомимся в этом параграфе.

В начале этой главы было сказано, что утверждение «В городе N живёт более 2 миллионов человек» нельзя считать логическим высказыванием, поскольку непонятно, о каком городе идёт речь. В этом предложении содержится некоторое утверждение, зависящее от N; если вместо N подставить название города, можно будет определить, истинно оно или ложно. Такое утверждение, зависящее от переменной, называют **логической функцией** или предикатом.

Предикат (от лат. *praedicatum* — заявленное, упомянутое, сказанное) — это утверждение, содержащее переменные.

Предикаты часто обозначаются буквой P, например:

$P(N)$ = «В городе N живёт более 2 миллионов человек».

Если мы задаём конкретные значения переменных, предикат превращается в логическое высказывание. Например, для предиката $P(N)$ мы получим истинное высказывание для N = «Москва» и ложное — для N = «Якутск».

Предикат, зависящий от одной переменной, — это **свойство** объекта. Например, только что рассмотренный предикат $P(N)$ характеризует свойство города. Вот ещё примеры **предикатов-свойств**:

Простое(x) = «x — простое число»;
Студент(x) = «x — студент»;
Спит(x) = «x всегда спит на уроке».

Предикаты могут зависеть от нескольких переменных, например:

Больше(x, y) = «x больше y»;
Живёт(x, y) = «x живёт в городе y»;
Любит(x, y) = «x любит y».

Это **предикаты-отношения**, они определяют связь между двумя объектами.

Предикаты нередко используются для того, чтобы задать множество, не перечисляя все его элементы. Так множество положительных чисел может быть задано предикатом, который принимает истинное значение для положительных чисел и ложное для остальных: $P(x) = (x > 0)$. Множество пар чисел, сумма которых равна 1, задается предикатом $P(x, y) = (x + y = 1)$, который зависит от двух переменных.

Существуют предикаты, которые справедливы (истинны) для всех допустимых значений переменных. Например, это предикат $P(x) = (x^2 \geq 0)$, определённый на множестве всех вещественных чисел. В таком случае используют запись $\forall x P(x)$, это означает: «При любом x предикат $P(x)$ справедлив». Знак $\forall$ — это буква «А», повёрнутая «вверх ногами» (от англ. *all* — все); он обозначает любой», «всякий», «для любого», «для всех». Символ $\forall$ называют **квантором всеобщности**.

Квантор (от лат. *quantum* — сколько) — это знак или выражение, обозначающее количество.

Выражения «любой», «для всех» и т. п. также можно считать кванторами, они равносильны знаку $\forall$.

Кванторы широко применяются в математике. Например, для натуральных n справедлива запись:

$$\forall n: \quad 1 + 2 + \ldots + n = \frac{n(n+1)}{2}.$$

Часто используют ещё один квантор — **квантор существования** $\exists$ (зеркальное отражение буквы «Е», от англ. *exist* — существовать). Знак $\exists$ означает «существует», «хотя бы один». Например, если $P(x) = (x - 5 > 0)$, то можно записать $\exists x P(x)$, что означает «Существует x, такой что $x - 5 > 0$». *Это уже высказывание, а не предикат*, потому что можно установить его истинность. Высказывание $\exists x P(x)$ — истинно, так как существует x, удовлетворяющий данному условию, например $x = 6$. Запись $\forall x P(x)$ — это тоже *высказывание*, но оно ложно, потому что неравенство $x - 5 > 0$ верно не для всех x.

Логическое выражение может включать несколько кванторов. Например, фразу «Для любого x существует y, такой что $x + y = 0$» можно записать как $\forall x \exists y (x + y = 0)$. Это утверждение истинно (на множестве чисел), потому что для любого x существует $-x$, число с обратным знаком. Переставлять местами кванторы нельзя, это меняет смысл выражения. Например, высказывание $\exists y \forall x (x + y = 0)$ означает: «Существует такое значение y, что для любого x выполняется равенство $x + y = 0$», это ложное высказывание.

Теперь давайте вернёмся к Сократу, точнее, к двум высказываниям, приведённым в начале параграфа. Как записать утверждение «Все люди смертны»? Можно сказать иначе: «Для любого x верно: если x — человек, то x смертен». Вспоминаем, что связка «если..., то» записывается как импликация, а выражение «для любого x» — в виде квантора $\forall x$. Поэтому получаем:

$$\forall x (P(x) \to Q(x)),$$

где $P(x)$ = «x — человек», $Q(x)$ = «x — смертен». Так как утверждение $P(x) \to Q(x)$ верно для любого x, оно также верно при подстановке x = Сократ:

$$P(\text{Сократ}) \to Q(\text{Сократ}) = 1.$$

Поскольку Сократ — человек, то $P(\text{Сократ}) = 1$. Поэтому с помощью таблицы истинности для импликации мы находим, что $Q(\text{Сократ}) = 1$, т. е. «Сократ смертен».

Если построить отрицание для высказывания с квантором $\forall$ или $\exists$, мы увидим, что один квантор заменяет другой. Например, отрицание высказывания $\forall x P(x)$ («Неверно, что для любого x выполняется $P(x)$») можно сформулировать так: «Существует хотя бы один x, для которого не выполняется $P(x)$» и может быть за-

писано в виде $\exists x\overline{P(x)}$. Здесь, как и раньше, черта сверху обозначает отрицание. Таким образом,

$$\overline{\forall x P(x)} = \exists x \overline{P(x)}.$$

Аналогично можно показать, что $\overline{\exists x P(x)} = \forall x \overline{P(x)}$.

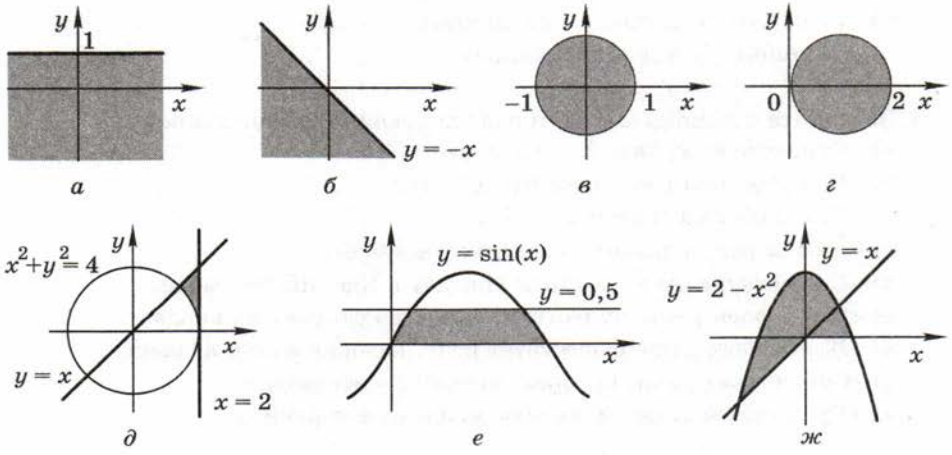
Где можно использовать язык предикатов? Самая подходящая для этого область информатики — системы искусственного интеллекта, в которых моделируется человеческое мышление. В таких системах часто применяется язык логического программирования Пролог, в котором программа представляет собой набор данных и правила вывода новых результатов из этих данных.

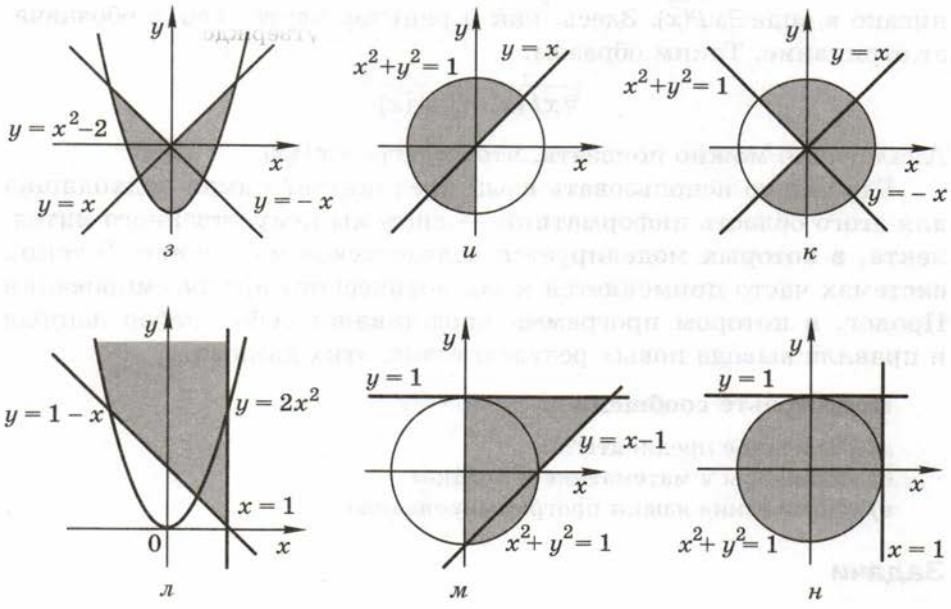
Подготовьте сообщение

а) «Что такое предикаты?»
б) «Кванторы в математике и логике»
в) «Логические языки программирования»

Задачи

1. Какие из следующих предложений являются предикатами (в заданиях а)–д) величины x и y — вещественные числа)?
   а) $x + y = 5$;
   б) $\exists x(x + y = 5)$;
   в) $\forall y \exists x(x + y = 5)$;
   г) $\sin^2 x + \cos^2 x = 1$;
   д) $x^2 + y^2 < 0$;
   е) «x работает в вузе»;
   ж) $\forall x$(«x — студент»);
   з) $\exists x$(«x — учитель y»).
2. Задайте с помощью предикатов множества точек, соответствующие заштрихованным областям на плоскости:





3. Поставьте в начале каждого предложения одно из слов: «все» или «не все».

а) «... окуни — рыбы».
б) «... рыбы умеют плавать».
в) «... реки впадают в моря».
г) «... моря солёные».
д) «... числа чётные».
е) «... ломаные состоят из отрезков».
ж) «... прямоугольники — квадраты».
з) «... кошки — млекопитающие».

4. Запишите с помощью кванторов следующие утверждения.

а) «Существует x, такой что $x > y$».
б) «Не существует x, такой что $x > y$».
в) «Для любого x имеем $x^2 > 1$».
г) «Любая река впадает в Каспийское море».
д) «Существует река, которая впадает в Каспийское море».
е) «Для любой реки существует море, в которое она впадает».
ж) «Для любого моря существует река, которая в него впадает».
з) «Существует река, которая впадает во все моря».
и) «Существует море, в которое впадают все реки».

*5. Запишите с помощью кванторов следующие утверждения:

а) «Некоторые школьники ходят в театр».

б) «Все кошки серые».

в) «Встречаются злые собаки».

г) «Все люди разные».

д) «Люди ошибаются».

е) «Никто не обращает на него внимания».

ж) «Ни одна фирма не обанкротилась».

з) «Все лебеди — белые или чёрные».

6. Запишите отрицание для следующих утверждений.

а) $\exists x(x^2 = 5)$;

б) $\exists x(x + y = 5)$;

в) $\forall y(x + y = 5)$;

г) $\forall y \exists x(x + y = 5)$;

д) «x работает в вузе»;

е) $\forall x$ («x — студент»);

ж) $\exists x$ («x — учитель y»);

з) $\exists x \forall y$ («x — учитель y»).

§ 24
Логические элементы компьютера

Простейшие элементы

В компьютерах все вычисления производятся с помощью **логических элементов** — электронных схем, выполняющих логические операции. Обозначения простейших элементов приводятся в таблице на рис. 3.21 (ГОСТ 2.743-91). Заметьте, что небольшой кружок на выходе (или на входе) обозначает операцию «НЕ» (отрицание, инверсию).

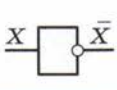
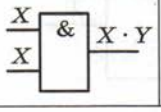
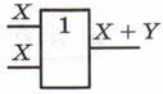
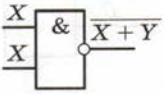
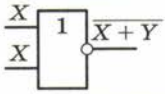
НЕ	И	ИЛИ	И – НЕ	ИЛИ – НЕ
				

Рис. 3.21

Может показаться, что для реализации сложных логических функций нужно много разных видов логических элементов. Однако, как мы видели в § 22, любую логическую функцию можно представить с помощью операций «НЕ», «И» и «ИЛИ» (такой набор элементов называется **полным**). Именно эта классическая «тройка» используется в книгах по логике, а также во всех языках программирования. Тем не менее инженеры часто предпочитают строить **логические схемы** на основе элементов «ИЛИ–НЕ». Как показано в § 19, эта функция (штрих Шеффера) позволяет реализовать операции «НЕ», «И» и «ИЛИ», а значит, и любую другую операцию.

Если нужно составить схему по известному логическому выражению, её начинают строить с конца. Находят операцию, которая будет выполняться последней, и ставят на выходе соответствующий логический элемент. Затем повторяют то же самое для сигналов, поступающих на вход этого элемента. В конце концов должны остаться только исходные сигналы — переменные в логическом выражении.

Составим схему, соответствующую выражению

$$X = \overline{A} \cdot B + A \cdot \overline{B} \cdot \overline{C}.$$

Последняя операция — это логическое сложение, поэтому на выходе схемы будет стоять элемент «ИЛИ» (рис. 3.22).

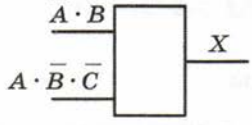


Рис. 3.22

Для того чтобы получить на первом входе $\overline{A} \cdot B$, нужно умножить $\overline{A}$ на B, поэтому добавляем элемент «И» (рис. 3.23).

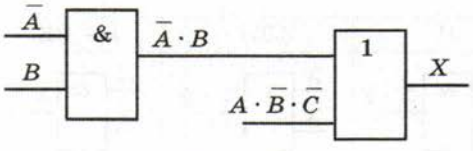


Рис. 3.23

Чтобы получить $\overline{A}$, ставим элемент «НЕ» (рис. 3.24).

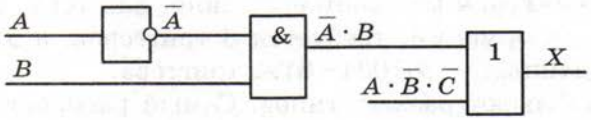


Рис. 3.24

Аналогично разбираем вторую ветку, которая поступает на второй вход элемента «ИЛИ» (рис. 3.25).

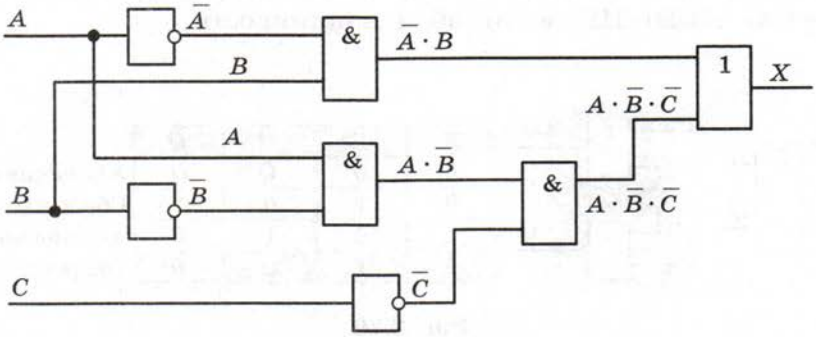


Рис. 3.25

Схема составлена, её входами являются исходные сигналы A, B и C, а выходом — X.

Триггер

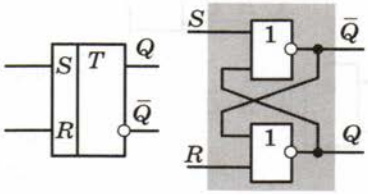
Слово **«триггер»** происходит от английского слова «trigger» — «защёлка» или спусковой крючок[1]. Так называют электронную схему, которая может находиться только в двух состояниях (их можно обозначить как 0 и 1) и способна почти мгновенно переходить из одного состояния в другое. Триггер изобрели независимо друг от друга М. А. Бонч-Бруевич и англичане У. Икклз и Ф. Джордан в 1918 г.

[1] В английском языке триггер называется flip-flop.

В современных компьютерах на основе триггеров строится быстродействующая оперативная память. Один триггер способен хранить один бит данных. Соответственно, для того, чтобы запомнить 1 байт информации, требуется 8 триггеров, а для хранения 1 килобайта данных — $8 \cdot 1024 = 8192$ триггера.

Триггеры бывают разных типов. Самый распространённый — это **RS-триггер**. Он имеет два входа, которые обозначаются как ***S*** (англ. *set* — установить) и ***R*** (англ. *reset* — сброс), и два выхода — Q и $\bar{Q}$, причём выходной сигнал $\bar{Q}$ является логическим отрицанием сигнала Q (если $Q = 1$, то $\bar{Q} = 0$, и наоборот).

RS-триггер можно построить на двух элементах «И–НЕ» или на двух элементах «ИЛИ–НЕ». На рисунке 3.26 показано условное обозначение RS-триггера, внутреннее устройство триггера на элементах «ИЛИ–НЕ» и его таблица истинности.



S	R	Q	$\bar{Q}$	
0	0	Q	$\bar{Q}$	Хранение бита
0	1	0	1	Сброс в 0
1	0	1	0	Установка в 1
1	1	0	0	Запрещено

Рис. 3.26

Триггер использует так называемые **обратные связи** — сигналы с выходов схем «ИЛИ–НЕ» поступают на вход соседней схемы. Именно это позволяет хранить информацию.

Построим таблицу истинности триггера. Начнем с варианта, когда $S = 0$ и $R = 1$. Элемент «ИЛИ–НЕ» в нижней части схемы можно заменить на последовательное соединение элементов «ИЛИ» и «НЕ». Независимо от второго входа, на выходе «ИЛИ» будет 1, а на выходе «НЕ» — ноль. Это значит, что $Q = 0$ (рис. 3.27).

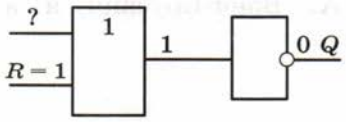


Рис. 3.27

Тогда на входе другого элемента «ИЛИ–НЕ» будут два нуля, а на выходе $\overline{Q}$ — единица (рис. 3.28).

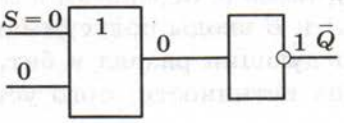


Рис. 3.28

Поскольку основным выходом считается Q, мы записали в триггер значение 0. Схема симметрична, поэтому легко догадаться, что при $S = 1$ и $R = 0$ мы запишем в триггер 1 ($Q = 1$).

Теперь рассмотрим случай, когда $S = 0$ и $R = 0$. На входе первого элемента «ИЛИ» будет сигнал $Q + 0 = Q$, поэтому на выходе $\overline{Q}$ останется его предыдущее значение (рис. 3.29).

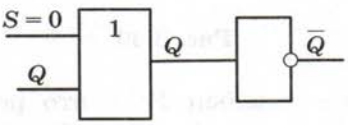


Рис. 3.29

Аналогично легко показать, что на выходе Q тоже остаётся его предыдущее значение. Это **режим хранения бита.**

Для случая $S = 1$ и $R = 1$ мы увидим, что оба выхода становятся равны нулю — в этом нет смысла, поэтому такой вариант запрещён.

Для хранения многоразрядных данных триггеры объединяются в единый блок, который называется **регистром**. Регистры (размером от 8 до 64 битов) используются во всех процессорах для временного хранения промежуточных результатов.

Над регистром, как над единым целым, можно производить ряд стандартных операций: сбрасывать (обнулять), заносить в него код и т. д. Часто регистры способны не просто хранить информацию, но и обрабатывать её. Например, существуют регистры-счётчики, которые подсчитывают количество импульсов, поступающих на вход.

Сумматор

Как следует из названия, сумматор предназначен для сложения (суммирования) двоичных чисел. Сначала рассмотрим более простой

элемент, который называют **полусумматором**. Он выполняет сложение двух битов с учетом того, что в результате может получиться двухразрядное число (с переносом в следующий разряд).

Обозначим через A и B входы полусумматора, а через P и S — выходы (перенос в следующий разряд и бит, остающийся в текущем разряде). Таблица истинности этого устройства показана на рис. 3.30.

A	B	P	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Рис. 3.30

Легко увидеть, что столбец P — это результат применения операции «И» ко входам A и B, а столбец S — результат операции «исключающее ИЛИ»:

$$P = A \cdot B, \qquad S = A \oplus B = \overline{A} \cdot B + A \cdot \overline{B}.$$

Формулу для S можно также записать в таком виде:

$$S = \overline{A} \cdot B + A \cdot \overline{B} = (A + B) \cdot (\overline{A} + \overline{B}) = (A + B) \cdot \overline{(A \cdot B)} = (A + B) \cdot \overline{P},$$

что позволяет построить полусумматор, используя всего четыре простейших элемента (рис. 3.31).

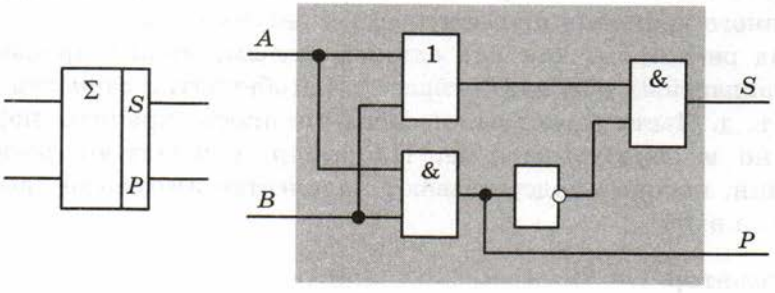


Рис. 3.31

Слева показано условное обозначение полусумматора, греческая буква Σ здесь (и в математике) обозначает сумму.

Полный **одноразрядный сумматор** учитывает также и третий бит C — перенос из предыдущего разряда. Сумматор имеет три входа и два выхода. Таблица истинности и обозначение сумматора показаны на рис. 3.32, 3.33.

A	B	C	P	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Рис. 3.32

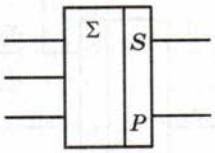


Рис. 3.33

Логические функции для выходов сумматора вы можете найти самостоятельно.

Сумматор можно построить с помощью двух полусумматоров и одного элемента «ИЛИ» (рис. 3.34).

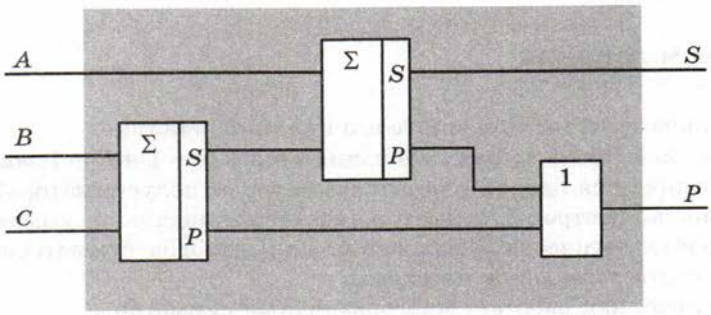


Рис. 3.34

Сначала складываются биты B и C, а затем к результату добавляется бит A. Перенос на выходе сумматора появляется тогда, когда любое из двух промежуточных сложений даёт перенос.

Для сложения многоразрядных чисел сумматоры объединяют в цепочку. При этом выход P одного сумматора (перенос в следующий разряд) соединяется с входом C следующего. На рисунке 3.35 показано, как складываются два трёхразрядных числа: $X = 110_2$ и $Y = 011_2$. Сумма $Z = 1001_2$ состоит из четырёх битов, поэтому на выходе последнего сумматора бит переноса будет равен 1.

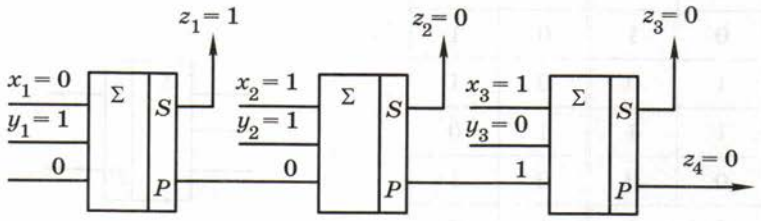


Рис. 3.35

Сложение начинается с самого младшего разряда. На вход первого сумматора подаются младшие биты исходных чисел, x_1 и y_1, а на третий вход — ноль (нет переноса из предыдущего разряда). Выход S первого сумматора — это младший бит результата, z_1, а его выход P (перенос) передаётся на вход второго сумматора и т. д. Выход P последнего из сумматоров представляет собой дополнительный разряд суммы, т. е. z_4.

Сумматор играет важную роль не только при сложении чисел, но при выполнении других арифметических действий. Фактически он является *основой арифметического устройства* современного компьютера.

Вопросы и задания

1. Что такое триггер? Объясните его принцип действия.
2. Почему для RS-триггера комбинация входов $S = 1$ и $R = 1$ запрещена?
3. Чем отличается одноразрядный сумматор от полусумматора?
4. Как можно построить сумматор с помощью двух полусумматоров?
5. Постройте логические выражения для выходов сумматора и нарисуйте соответствующие им схемы.
6. Объясните, как работает многоразрядный сумматор.
7. Что такое перенос? Как он используется в многоразрядном сумматоре?

Подготовьте сообщение

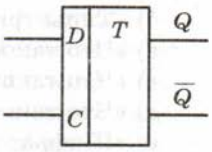
а) «Обозначения логических элементов в России и за рубежом»
б) «Типы триггеров»
в) «Что такое регистр?»
г) «Что такое мультиплексор?»
д) «Что такое демультиплексор?»
е) «Шифратор и дешифратор»

Задачи

1. Используя логические элементы, постройте схемы, соответствующие логическим выражениям:
   а) $X_1 = \overline{A} \cdot C + B \cdot \overline{C}$;
   б) $X_2 = A \cdot B + \overline{B} \cdot \overline{C}$;
   в) $X_3 = \overline{A} \cdot \overline{B} + B \cdot \overline{C}$.
2. Соревнования по поднятию тяжестей судит бригада из трёх человек, один из них старший. Лампочка «Вес взят» должна зажигаться, если проголосовали, по крайней мере, два судьи, причём один из них — старший. Предложите логическую схему, которая решала бы эту задачу.
3. В двухэтажном коттедже есть два выключателя, которые управляют освещением лестницы, один из них — на первом этаже, а второй — на втором. Каждый выключатель имеет два состояния, при нажатии на кнопку состояние изменяется. В исходный момент оба выключателя выключены. Когда человек заходит в неосвещённое здание, он нажимает кнопку выключателя на первом этаже, при этом должна загореться лампочка, освещающая всю лестницу. Поднявшись на второй этаж, он нажимает на кнопку второго выключателя, и лампочка должна погаснуть. Когда следом идет другой человек, он действует так же (хотя оба выключателя находятся в другом положении). Предложите логическую схему, которая решала бы эту задачу.
4. В самолёте есть три бака с горючим. Бортовой компьютер получает сигналы от датчиков уровня в каждом баке: если горючего в баке достаточно, то сигнал равен 0, если горючее кончилось — 1. Когда горючее заканчивается, по крайней мере, в двух баках, должна загореться лампочка «Тревога». Предложите логическую схему, которая решала бы эту задачу.
5. В парламенте некоторой страны выбирают спикера из трёх кандидатов. Каждый парламентарий должен нажать одну и только одну из трёх кнопок. Если он проголосовал правильно (нажал ровно одну кнопку), на пульте должна загореться зелёная лампочка. Предложите логическую схему, которая решала бы эту задачу.

6. Постройте RS-триггер на элементах «И–НЕ» и составьте его таблицу истинности.

*7. Постройте таблицу истинности и логическую схему *D*-триггера, который запоминает сигнал, поступающий на вход *D* (*англ.* data — данные), при подаче логической единицы на вход *C* (англ. *clock* — синхронизация). Для этого можно, например, немного изменить входную часть RS-триггера.



§ 25
Логические задачи

Метод рассуждений

Задача 1. Среди трёх приятелей (их зовут Сеня, Вася и Миша) один всегда говорит правду, второй говорит правду через раз, а третий всё время обманывает. Как-то раз они впервые прогуляли урок информатики. Директор школы вызвал их в свой кабинет для разговора. Сеня сказал: «Я всегда прогуливаю информатику. Не верьте тому, что скажет Вася». Вася сказал: «Я раньше не прогуливал этот предмет». Миша сказал: «Всё, что говорит Сеня, — правда». Директору стало всё понятно. Кто из них правдив, кто лгун, а кто говорит правду через раз?

Для решения используем **метод рассуждений**. Во-первых, есть точная информация, которая не подвергается сомнению: все трое прогуляли урок информатики в первый раз.

Запишем высказывания мальчиков:

Сеня:	1. Я всегда прогуливаю информатику.
	2. Вася сейчас скажет неправду.
Вася:	1. Я раньше не прогуливал информатику.
Миша:	1. Сеня говорит правду.

Известно, что один из них говорит правду всегда, второй через раз, а третий всё время лжёт. Отметим, что если у нас есть только одно высказывание «полулжеца», оно может быть как истинным, так и ложным.

Сопоставив первое высказывание Сени и высказывание Васи с точной информацией, сразу определяем, что тут Сеня соврал, а Вася сказал правду. Это значит, что второе высказывание Сени тоже неверно, поэтому Сеня всегда лжёт.

Тогда один из оставшихся, Вася или Миша, правдив, а второй говорит правду через раз. Мишино высказывание неверно, поскольку мы уже определили, что Сеня всегда лжёт; это значит, что Миша не всегда говорит правду, он — «полулжец». Тогда получается, что Вася правдив.

Табличный метод

Задача 2. Перед началом турнира по шахматам болельщики высказали следующие предположения по поводу результатов:

А) Максим победит, Борис будет вторым;
Б) Борис займёт третье место, а Коля — первое;
В) Максим будет последним, а Дима — первым.

Когда соревнования закончились, оказалось, что каждый из болельщиков был прав только в одном из своих прогнозов. Как распределились призовые места, если каждое место занял ровно один участник?

Запишем высказывания трёх болельщиков в форме таблицы (заголовок строки обозначает место в турнирной таблице):

	А	Б	В
1	Максим?	Коля?	Дима?
2	Борис?		
3		Борис?	
4			Максим?

Начнём «раскручивать» эту таблицу с той строчки, где больше всего информации, в данном случае — с первой. Предположим, что Максим действительно занял первое место, как и сказал болельщик «А». В этом случае «В» ошибся, поставив на первое место Диму. Тогда получается, что второй прогноз болельщика «В» верен, и Максим — последний.

Так как мы предполагали, что Максим занял первое место, получается противоречие. Следовательно, первый прогноз «А» не

сбылся. Но тогда должен быть верен его второй прогноз, и Борис занял второе место. При этом он не мог занять ещё и третье место, поэтому первый прогноз болельщика «Б» неверный, а верен его второй прогноз: Коля — первый.

В этом случае Дима не может быть первым, поэтому верен первый прогноз «В»: Максим — последний. Диме осталось единственное свободное третье место.

	А	Б	В
1		Коля	
2	Борис		
3			
4			Максим

В результате места распределились так: I — Коля, II — Борис, III — Дима и IV — Максим.

Задача 3. На одной улице стоят в ряд четыре дома, в каждом из них живёт по одному человеку. Их зовут Василий, Семён, Геннадий и Иван. Известно, что все они имеют разные профессии: скрипач, столяр, охотник и врач. Известно, что:

(1) Столяр живёт правее охотника.
(2) Врач живёт левее охотника.
(3) Скрипач живёт с краю.
(4) Скрипач живёт рядом с врачом.
(5) Семён не скрипач и не живёт рядом со скрипачом.
(6) Иван живёт рядом с охотником.
(7) Василий живёт правее врача.
(8) Василий живёт через дом от Ивана.

Определите, кто где живёт.

Из условий (1) и (2) следует, что охотник живет не с краю, потому что справа от него живёт столяр, а слева — врач. Скрипач по условию (3) живёт с краю, он может жить как слева, так и справа от остальных:

скрипач?	врач	охотник	столяр	скрипач?

Согласно условию (4), скрипач живет рядом с врачом, поэтому он занимает крайний дом слева:

1	2	3	4
скрипач	врач	охотник	столяр

Профессии жильцов определили, остаётся разобраться с именами. Из условия (5) «Семён не скрипач и не живет рядом со скрипачом» следует, что Семён — охотник или столяр:

1	2	3	4
скрипач	врач	охотник	столяр
		Семён?	Семён?

Из условия (6) «Иван живёт рядом с охотником» следует, что он — врач или столяр:

1	2	3	4
скрипач	врач	охотник	столяр
		Семён?	Семён?
	Иван?		Иван?

Из условия (7) «Василий живет правее врача» определяем, что Василий — охотник или столяр:

1	2	3	4
скрипач	врач	охотник	столяр
		Семён?	Семён?
	Иван?		Иван?
		Василий?	Василий?

Согласно условию (8), «Василий живёт через дом от Ивана», поэтому Иван — врач, а Василий — столяр:

1	2	3	4
скрипач	врач	охотник	столяр
	Иван	Семён?	Василий

Тогда сразу получается, что Семён — охотник, а Геннадий должен занять оставшееся свободное место, он — скрипач:

1	2	3	4
скрипач	врач	охотник	столяр
Геннадий	Иван	Семён	Василий

Задача 4. Шесть приятелей, Саша, Петя, Витя, Дима, Миша и Кирилл, встретившись через 10 лет после окончания школы, выяснили, что двое из них живут в Москве, двое — в Санкт-Петербурге, а двое — в Перми. Известно, что:

(1) Витя ездит в гости к родственникам в Москву и Санкт-Петербург.
(2) Петя старше Саши.
(3) Дима и Миша летом были в Перми в командировке.
(4) Кирилл и Саша закончили университет в Санкт-Петербурге и уехали в другие города.
(5) Самый молодой из них живёт в Москве.
(6) Кирилл редко приезжает в Москву.
(7) Витя и Дима часто бывают в Санкт-Петербурге по работе.

Определите, кто где живёт.

Составим таблицу, где каждая строка соответствует городу, а столбец — человеку:

	Саша	Петя	Витя	Дима	Миша	Кирилл
Москва						
Санкт-Петербург						
Пермь						

Единица в таблице будет обозначать, что человек живёт в данном городе, а ноль — что не живёт. По условию в каждом городе живут ровно 2 человека, каждый живёт только в одном городе. Поэтому в каждой строке должно быть две единицы, а в каждом столбце — одна.

Из условия (1) следует, что Витя живёт в Перми:

	Саша	Петя	Витя	Дима	Миша	Кирилл
Москва			0			
Санкт-Петербург			0			
Пермь			1			

Из (2) и (5) находим, что Петя живёт не в Москве. Кроме того, как следует из (6), Кирилл — тоже не москвич.

	Саша	Петя	Витя	Дима	Миша	Кирилл
Москва		0	0			0
Санкт-Петербург			0			
Пермь			1			

Согласно (3), Дима и Миша живут не в Перми:

	Саша	Петя	Витя	Дима	Миша	Кирилл
Москва		0	0			0
Санкт-Петербург			0			
Пермь			1	0	0	

Из условия (4) делаем вывод, что Кирилл и Саша живут не в Санкт-Петербурге, отсюда сразу следует, что Кирилл живёт в Перми. Двух пермяков мы уже определили, поэтому Саша и Петя живут не в Перми:

	Саша	Петя	Витя	Дима	Миша	Кирилл
Москва		0	0			0
Санкт-Петербург	0		0			0
Пермь	0	0	1	0	0	1

Далее определяем из таблицы, что Саша — москвич, а Петя живёт в Санкт-Петербурге:

	Саша	Петя	Витя	Дима	Миша	Кирилл
Москва	1	0	0			0
Санкт-Петербург	0	1	0			1
Пермь	0	0	1	0	0	1

По условию (7), Витя и Дима — не петербуржцы, поэтому в Петербурге живёт Миша. Тогда Дима живёт в Москве:

	Саша	Петя	Витя	Дима	Миша	Кирилл
Москва	1	0	0	1	0	0
Санкт-Петербург	0	1	0	0	1	1
Пермь	0	0	1	0	0	1

Таким образом, Саша и Дима живут в Москве, Петя и Миша — в Санкт-Петербурге, а Витя и Кирилл — в Перми.

Использование алгебры логики

Когда в условии задачи встречаются сложные логические высказывания, удобно использовать методы алгебры логики. Покажем этот подход на примерах.

Задача 5. Следующие два высказывания истинны:

1. Неверно, что если корабль А вышел в море, то корабль С — нет.
2. В море вышел корабль В или корабль С, но не оба вместе.

Определить, какие корабли вышли в море.

Введём три высказывания: A — «Корабль A вышел в море»; B — «Корабль В вышел в море»; C — «Корабль С вышел в море». Вспомним, что связка «если..., то» в логических выражениях заменяется импликацией, поэтому фразу «Если корабль A вышел

в море, то корабль C — нет» можно записать как $A \to \overline{C} = 1$. Но в условии сказано, что это утверждение неверно, поэтому:

$$A \to \overline{C} = 0, \text{ или } \overline{A \to \overline{C}} = 1.$$

Второе условие — это операция «исключающее ИЛИ», т. е. $B \oplus C = 1$. Оба условия истинны одновременно, т. е. их логическое произведение (И) тоже истинно:

$$\overline{(A \to \overline{C})} \cdot (B \oplus C) = 1.$$

Нам нужно решить это уравнение и найти неизвестные A, B и C. Для этого выразим импликацию и операцию «исключающее ИЛИ» через базовый набор логических операций (НЕ, И, ИЛИ), а затем раскроем инверсию сложного выражения с помощью закона де Моргана:

$$\overline{(A \to \overline{C})} \cdot (B \oplus C) = \overline{(\overline{A} + \overline{C})} \cdot (B \cdot \overline{C} + \overline{B} \cdot C) = A \cdot C \cdot (B \cdot \overline{C} + \overline{B} \cdot C) = 1.$$

В последнем выражении раскроем скобки и учтём, что $C \cdot \overline{C} = 0$ и $C \cdot C = C$. Получим:

$$A \cdot \overline{B} \cdot C = 1.$$

Это уравнение имеет единственное решение: $A = 1$, $B = 0$ и $C = 1$. Это значит, что в море вышли корабли A и C.

Выше был показан общий подход к решению подобных задач, однако эту конкретную задачу можно решить намного проще. Как вы знаете, импликация $A \to B$ ложна только при $A = 1$ и $B = 0$. Поэтому из условия $A \to \overline{C} = 0$ сразу следует, что $A = C = 1$. Теперь остаётся только применить второе условие $B \oplus C = 1$, которое при $C = 1$ даёт $B = 0$, и получаем тот же ответ, что и раньше.

Задача 6. На вопрос «Кто из твоих учеников изучал логику?» учитель ответил: «Если логику изучал Андрей, то изучал и Борис. Однако неверно, что если изучал Семён, то изучал и Борис». Кто же изучал логику?

Обозначим переменными высказывания: A — «Логику изучал Андрей»; B — «Логику изучал Борис» и C — «Логику изучал Семён». Оба высказывания учителя можно записать в виде импликаций:

«Если логику изучал Андрей, то изучал и Борис». $A \to B = 1$
«Неверно, что если изучал Семён, то изучал и Борис». $C \to B = 0$

Дальше есть два варианта решения. Во-первых, можно поступить так же, как и в предыдущей задаче: применить операцию

«НЕ» ко второму высказыванию и составить уравнение с помощью логического произведения:

$$(A \to B) \cdot \overline{(C \to B)} = 1.$$

Теперь представляем импликацию через базовые операции и применяем закон де Моргана

$$(\overline{A} + B) \cdot \overline{(\overline{C} + B)} = (\overline{A} + B) \cdot C \cdot \overline{B} = \overline{A} \cdot C \cdot \overline{B} = 1.$$

Это уравнение имеет единственное решение: $A = 0$, $B = 0$ и $C = 1$. Значит, логику изучал только Семён.

Можно поступить иначе, вспомнив, что импликация ложна только в том случае, когда первое высказывание истинно, а второе ложно. Поэтому из условия $C \to B = 0$ сразу следует, что $B = 0$ и $C = 1$. Тогда первое условие, $A \to B = A \to 0 = 1$ сразу даёт $A = 0$.

Подготовьте сообщение

а) «Задача Эйнштейна»
б) «Задачи о лжецах»
в) «Задачи о шляпах»
г) «Задачи о двух городах»

Задачи

1. Три школьника — Миша, Коля и Сергей, остававшиеся в классе на перемене, были вызваны к директору по поводу разбитого в это время окна в кабинете. На вопрос директора о том, кто это сделал, мальчики ответили следующее:

   Миша: «Я не разбивал окно, и Коля тоже».
   Коля: «Миша не разбивал окно, это Сергей разбил футбольным мячом!»
   Сергей: «Я не делал этого, стекло разбил Миша».

   Выяснилось, что один из ребят сказал правду, второй в одной части заявления солгал, а другое его высказывание истинно, а третий оба раза солгал. Кто разбил стекло в классе?
2. В финал соревнований по настольному теннису вышли Наташа, Маша, Люда и Рита. Болельщики высказали свои предположения о распределении мест в дальнейших состязаниях. Один считает, что первой будет Наташа, а Маша — второй. Другой болельщик на второе место прочит Люду, а Рита, по его мнению, займёт четвёртое место. Третий считает, что Рита займёт третье место, а Наташа будет второй. Когда соревнования закончились, оказалось, что каждый из болельщиков был прав только в одном из своих прогнозов. Как распределились места?

3. На одной улице стоят в ряд четыре дома, в каждом из них живёт по одному человеку. Их зовут Алексей, Егор, Виктор и Михаил. Известно, что все они имеют разные профессии: рыбак, пчеловод, фермер и ветеринар. Известно, что:

   (1) Фермер живёт правее пчеловода.
   (2) Рыбак живёт правее фермера.
   (3) Ветеринар живёт рядом с рыбаком.
   (4) Рыбак живёт через дом от пчеловода.
   (5) Алексей живёт правее фермера.
   (6) Виктор не пчеловод.
   (7) Егор живёт рядом с рыбаком.
   (8) Виктор живёт правее Алексея.

   Определите, кто где живёт.

4. Дочерей Василия Лоханкина зовут Даша, Анфиса и Лариса. У них разные профессии, и они живут в разных городах: одна в Ростове, вторая — в Париже и третья — в Москве. Известно, что:

   (1) Даша живёт не в Париже, а Лариса — не в Ростове.
   (2) Парижанка не актриса.
   (3) В Ростове живёт певица.
   (4) Лариса не балерина.

   Определите, где живёт каждая из дочерей и чем занимается.

5. В состав экспедиции входят Михаил, Сергей и Виктор. На обсуждении распределения обязанностей с руководителем проекта были высказаны предположения, что командиром будет назначен Михаил, Сергей не будет механиком, а Виктор будет утвержден радистом, но командиром точно не будет. Позже выяснилось, что только одно из этих четырёх утверждений оказалось верным. Как распределились должности?

6. В ходе заседания суда выяснилось, что:

   (1) Если Аськин не виновен или Баськин виновен, то виновен Сенькин.
   (2) Если Аськин не виновен, то Сенькин не виновен.

   Что можно сказать о виновности Аськина, Баськина и Сенькина?

7. Аськин, Баськин и Васькин стали свидетелями ограбления банка. Во время расследования Аськин сказал, что взломщики приехали на синей «Тойоте». Баськин считает, что это был красный «BMW», а Васькин утверждает, что это был «Форд-Фокус», но не синий. Выяснилось, что каждый из них назвал неправильно либо марку, либо цвет машины. На каком автомобиле приехали преступники?

Практические работы к главе 3

Работа № 7 «Тренажёр "Логика"»
Работа № 8 «Исследование запросов для поисковых систем»

ЭОР к главе 3 на сайте ФЦИОР (http//fcior.edu.ru)

- Высказывание. Простые и сложные высказывания
- Основные логические операции
- Теория множеств
- Логические законы и правила преобразования логических выражений
- Построение отрицания к простым высказываниям, записанным на русском языке
- Построение отрицания к сложным высказываниям, записанным на русском языке
- Решение логических задач
- Сумматор двоичных чисел

Самое важное в главе 3

- Логическое выражение может принимать два значения: «истина» и «ложь». Если обозначить эти значения символами 0 и 1, то получится, что с помощью логических операций можно описать правила обработки двоичных кодов.
- Все вычисления в компьютерах выполняются с помощью логических операций с двоичными кодами данных.
- Для определения логической операции используют таблицы истинности. Таблица истинности однозначно определяет некоторую логическую функцию — правило преобразования входных данных в результат.
- Используя только логические операции «И», «ИЛИ» и «НЕ», можно записать любую логическую функцию.
- Любой логической функции соответствует множество логических выражений. Используя законы алгебры логики, логические выражения можно преобразовывать и упрощать.
- Предикат — это утверждение, содержащее переменные. Если значения всех переменных заданы, предикат превращается в логическое высказывание.
- Триггер — это логическая схема, которая способна запоминать один бит данных.
- Сумматор — это логическая схема для сложения двоичных чисел.

Глава 4
Компьютерная арифметика

§ 26
Особенности представления чисел в компьютере

На уроках математики вы никогда не обсуждали, как хранятся числа. Математика — это теоретическая наука, для которой совершенно не важно, записаны они на маленьком или большом листе бумаги, зафиксированы с помощью счётных палочек, счётов, или внутри полупроводниковой схемы. Поэтому число в математике может состоять из любого количества цифр, которое требуется в решаемой задаче.

В то же время инженеры, разрабатывающие компьютер, должны спроектировать реальное устройство из вполне определённого количества деталей. Поэтому число разрядов, отведённых для хранения каждого числа, ограничено, и точность вычислений тоже ограничена. Из-за этого при компьютерных расчётах могут возникать достаточно серьёзные проблемы. Например, сумма двух положительных чисел может получиться отрицательной, а выражение $A + B$ может совпадать с A при ненулевом B. В этой главе мы рассмотрим важные особенности компьютерной арифметики, которые нужно учитывать при обработке данных. В первую очередь, они связаны с тем, как размещаются целые и вещественные числа в памяти компьютера.

Предельные значения чисел

Как вы уже поняли, числа, хранящиеся в компьютере, не могут быть сколь угодно большими и имеют некоторые предельные значения. Представим себе некоторое вычислительное устройство, которое работает с четырехразрядными неотрицательными целыми десятичными числами (рис. 4.1). Для вывода чисел используется четырёхразрядный индикатор, на котором можно отобразить числа от 0 (все разряды числа минимальны) до 9999 (все разряды максимальны) — рис. 4.2.

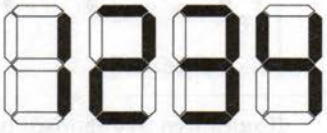


Рис. 4.1

Диапазон от  до 

Рис. 4.2

Вывести на такой индикатор число 10 000 невозможно: не хватает технического устройства для пятого разряда. Такая «аварийная» ситуация называется переполнением разрядной сетки или просто переполнением (англ. *overflow* — переполнение «сверху»).

Переполнение разрядной сетки — это ситуация, когда число, которое требуется сохранить, не умещается в имеющемся количестве разрядов вычислительного устройства.

В нашем примере переполнение возникает при значениях, больших $9999 = 10^4 - 1$, где 4 — это количество разрядов. В общем случае, если в системе счисления с основанием B для записи числа используется K разрядов, максимальное допустимое число $C_{\max}$ вычисляется по аналогичной формуле[1]

$$C_{\max} = B^K - 1.$$

Именно эта формула для $B = 2$ неоднократно применялась в главе 2.

Подчеркнём, что переполнение никак не связано с системой счисления: оно вызвано *ограниченным количеством разрядов* устройства и не зависит от количества возможных значений в каждом из этих разрядов.

Рассмотрим теперь, что получится, если наше устройство будет работать не только с целыми, но и с дробными числами. Пусть, например, один из четырёх разрядов относится к целой части числа, а остальные три — к дробной (рис. 4.3). Конечно, эффект переполнения сохранится и здесь: максимально допустимое число равно 9,999. Кроме того, дробная



Рис. 4.3

[1] Докажите эту формулу самостоятельно, например, подсчитав количество всех возможных комбинаций значений цифр в K разрядах.

часть числа тоже ограничена, поэтому любое число, имеющее более трёх цифр после запятой, не может быть представлено точно: младшие цифры придётся отбрасывать (или округлять).

Не все вещественные числа могут быть представлены в компьютере точно.



Рис. 4.4

При ограниченном числе разрядов дробной части существует некоторое минимальное ненулевое значение $C_{\min}$, которое можно записать на данном индикаторе (в нашем примере это 0,001, рис. 4.4). В общем случае, если число записано в системе счисления с основанием B и для хранения дробной части числа используется F разрядов, имеем

$$C_{\min} = B^{-F}.$$

Любое значение, меньшее чем $C_{\min}$, неотличимо от нуля. Такой эффект принято называть *антипереполнением* (англ. *underflow* — переполнение «снизу»).



Рис. 4.5

Кроме того, два дробных числа, отличающиеся менее чем на $C_{\min}$, для компьютера неразличимы. Например, 1,3212 и 1,3214 на нашем индикаторе выглядят совершенно одинаково (рис. 4.5).

Дополнительная погрешность появляется при переводе дробных чисел из десятичной системы счисления в двоичную. При этом даже некоторые «круглые» числа (например, 0,2) в памяти компьютера представлены неточно, потому что в двоичной системе они записываются как бесконечные дроби и их приходится округлять до заданного числа разрядов.

Так как вещественные числа хранятся в памяти приближённо, сравнивать их (особенно если они являются результатами сложных расчётов) необходимо с большой осторожностью. Пусть при вычислениях на компьютере получили $X = 10^{-6}$ и $Y = 10^{6}$. Дробное значение X будет неточным, и произведение $X \cdot Y$ может незначительно отличаться от 1. Поэтому при сравнении вещественных чисел в компьютере условие «равно» использовать не рекомендуется. В таких случаях числа считаются равными, если

их разность достаточно мала по модулю. В данном примере нужно проверять условие $|1 - X \cdot Y| < \varepsilon$, где ε — малая величина, которая задаёт нужную точность вычислений. К счастью, для большинства практических задач достаточно взять ε порядка $10^{-2} \ldots 10^{-4}$, а ошибка компьютерных расчётов обычно значительно меньше[1] (не более 10^{-7}).

Введение разряда для знака числа не меняет сделанных выше выводов, только вместо нулевого минимального значения появляется отрицательное, которое зависит от разрядности (оно равно –9999 в первом из обсуждаемых примеров).

Различие между вещественными и целыми числами

Существуют величины, которые по своей природе могут принимать только целые значения, например счётчики повторений каких-то действий, количество людей или предметов, координаты пикселей на экране и т. п. Кроме того, как показано в главе 2, кодирование нечисловых видов данных (текста, изображений, звука) сводится именно к целым числам.

Чтобы сразу исключить все возможные проблемы, связанные с неточностью представления в памяти вещественных чисел, целочисленные данные кодируются в компьютерах особым образом.



Целые и вещественные числа в компьютере хранятся и обрабатываются по-разному.

Операции с целыми числами, как правило, выполняются значительно быстрее, чем с вещественными. Не случайно в ядре современных процессоров реализованы только целочисленные арифметические действия, а для вещественной арифметики используется специализированный встроенный блок — *математический сопроцессор*.

Кроме того, использование целых типов данных позволяет экономить компьютерную память. Например, целые числа в интервале от 0 до 255 в языке Паскаль можно хранить в перемен-

[1] Тем не менее встречаются ситуации, когда вычислительные трудности все же возникают: классический пример — разность близких по значению десятичных дробей, отличающихся в последних значащих цифрах.

ных типа *byte*, которые занимают всего один байт в памяти. В то же время самое «короткое» вещественное число (типа *single*) требует четырёх байтов памяти.

Наконец, только для целых чисел определены операции деления нацело и нахождения остатка. В некоторых задачах они удобнее, чем простое деление с получением дробного (к тому же не совсем точного) результата: например, без них не обойтись при вычислении суммы цифр какого-то числа.

Таким образом, для всех величин, которые не могут иметь дробных значений, нужно использовать целочисленные типы данных.

Дискретность представления чисел

Из § 7 вы знаете, что существует непрерывное и дискретное представление информации. Их принципиальное различие состоит в том, что дискретная величина может принимать конечное количество различных значений в заданном интервале, а непрерывная имеет бесконечно много возможных значений. Для нашего обсуждения важно, что:

- целые числа дискретны;
- вещественные (действительные, дробные) числа непрерывны;
- современный компьютер работает только с дискретными данными.

Таким образом, для хранения вещественных чисел в памяти компьютера нужно выполнить *дискретизацию* — записать непрерывную величину в дискретной форме. При этом может происходить искажение данных, поэтому большинство трудностей в компьютерной арифметике (антипереполнение, приближённость представления дробной части и др.) связано именно с кодированием дробных чисел.

Программное повышение точности вычислений

Современные модели процессоров Intel «умеют» обрабатывать 8-, 16-, 32- и 64-разрядные двоичные целые числа, а также (в математическом сопроцессоре) 32-, 64- и 80-разрядные вещественные числа. Для большинства практических задач такой разрядности вполне достаточно. Если для каких-либо особо точных расчетов требуется повысить разрядность вычислений, это можно сделать программно. Например, можно считать, что четыре

последовательно хранящихся целых 64-разрядных числа — это единое «длинное» число, и написать программу обработки таких «удлинённых» чисел. Очень удобно хранить числа в виде последовательности десятичных цифр[1], правда, программы, выполняющие обработку таких чисел, получаются сложными и медленными.

Использование этих и других программных методов позволяет увеличить разрядность обрабатываемых чисел по сравнению с аппаратной разрядностью компьютера. Однако ограничение разрядности (и связанный с ним эффект переполнения) все равно остаётся: в программу заложено конкретное число разрядов, да и объём памяти компьютера конечен.

Вопросы и задания

1. Чем отличается компьютерная арифметика от «обычной»? Почему?
2. Почему диапазон чисел в компьютере ограничен? Связано ли это с двоичностью компьютерной арифметики?
3. Что такое переполнение разрядной сетки?
4. Какие проблемы появляются при ограниченном числе разрядов в дробной части?
5. Что называется антипереполнением? Что, по-вашему, опаснее для вычислений — переполнение или антипереполнение?

*6. Может ли антипереполнение сделать невозможными дальнейшие вычисления?

7. Сколько битов информации несёт знаковый разряд?
8. Приведите примеры величин, которые по своему смыслу могут иметь только целые значения.
9. Какая математическая операция между двумя целыми числами может дать в результате нецелое число?
10. Чем различается деление для целых и вещественных чисел?
11. Какие преимущества даёт разделение в компьютере целых и вещественных (дробных) чисел?
12. Вспомните определение дискретных и непрерывных величин. Какие множества чисел в математике дискретны, а какие — нет? Ответ обоснуйте.
13. Объясните, почему ограниченность разрядов дробной части приводит к нарушению свойства непрерывности.

---

[1] Такие задачи часто даются на школьных олимпиадах по информатике; для них даже придумано специальное название: «длинная» арифметика.

14. Можно ли организовать вычисления с разрядностью, превышающей аппаратную разрядность компьютера? Попробуйте предложить способы решения этой задачи.

Подготовьте сообщение

а) «Проблемы вычислений с вещественными данными»
б) «Длинная арифметика»

Задачи

1. Вычислите максимальное целое значение для 8-разрядного двоичного числа и 3-разрядного десятичного (считать, что значения не бывают отрицательными). Какое из них оказалось больше?
2. Вычислите максимальное целое положительное значение для 16- и 32-разрядных двоичных чисел.
3. Пользуясь калькулятором, вычислите границу антипереполнения для чисел с 16 двоичными разрядами в дробной части. Напишите два близких дробных числа, которые для полученного значения окажутся неразличимыми.

*4. Вычислите минимально возможное отрицательное значение для 16-разрядных двоичных чисел (учесть, что один из двоичных разрядов является знаковым).

*5. Придумайте простую вычислительную задачу, в которой для хранения результата не хватает 16 двоичных разрядов.

§ 27
Хранение в памяти целых чисел

Целые числа без знака

Беззнаковые (англ. *unsigned*) типы данных, т. е. величины, не имеющие отрицательных значений, широко используются в вычислительной технике. Дело в том, что в задачах, решаемых на компьютерах, есть много таких значений: всевозможные счётчики (количество повторений циклов, число параметров в списке или символов в тексте), количество людей или предметов и др.

Чтобы закодировать целое число без знака, достаточно перевести его в двоичную систему счисления (см. § 11) и дополнить

слева нулями до нужной разрядности. Например, число 28 записывается в 8-разрядную ячейку памяти так:

0001 1100

Это же число в 16-разрядном представлении будет иметь слева ещё 8 нулей. Восьмиразрядные коды некоторых характерных чисел приведены в табл. 4.1.

Таблица 4.1

X_{10}	0	1	...	127	128	129	...	255
X_{16}	00	01	...	7F	80	81	...	FF
X_2	0000 0000	0000 0001	...	0111 1111	1000 0000	1000 0001	...	1111 1111

Минимальное значение для беззнаковых целых чисел всегда равно 0 (все разряды нулевые), а максимальное число $X_{max} = 2^K - 1$ состоит из всех единиц и определяется разрядностью (количеством битов) K (табл. 4.2).

Таблица 4.2

K, битов	8	16	32	64
$X_{max} = 2^K - 1$	255	65 535	4 294 967 295	18 446 744 073 709 551 615

Возникает вопрос: что будет, если увеличить максимальное число в K-битной ячейке на единицу? Рассмотрим случай $K = 8$ и попытаемся прибавить единицу к числу $255_{10} = 1111\ 1111_2$. Добавляя дополнительный бит слева, получим:

$$\begin{array}{rcl} & 0 & 1111\ 1111 \\ + & 0 & 0000\ 0001 \\ \hline & 1 & 0000\ 0000 \end{array}$$

Отбросив несуществующий дополнительный разряд[1], получаем 255 + 1 = 0. Как ни странно, именно это произойдёт в реальном компьютере. Говорят, что при K разрядах арифметика выполняется **по модулю 2^K**, т. е. при $K = 8$ имеем[2]:

$$(255 + 1) \bmod 256 = 256 \bmod 256 = 0.$$

---

[1] На самом деле, для того чтобы обнаружить факт переполнения, этот разряд сохраняется в специальном управляющем бите процессора, который называется битом переноса.

[2] Здесь запись $a \bmod b$ обозначает остаток от деления a на b.

Вместе с тем, вычитая единицу из минимального значения 0, к которому добавлен старший разряд за пределами 8-битной ячейки, получим $1111\ 1111_2 = 255_{10}$ (проверьте это самостоятельно).

Можно заметить, что при многократном увеличении числа на единицу мы доходим до максимального значения и скачком возвращаемся к минимальному. При вычитании единицы получается обратная картина — дойдя до минимума (нуля), мы сразу перескакиваем на максимум (255). Поэтому для изображения допустимого диапазона чисел лучше подходит не отрезок числовой оси (как в математике), а окружность (рис. 4.6).

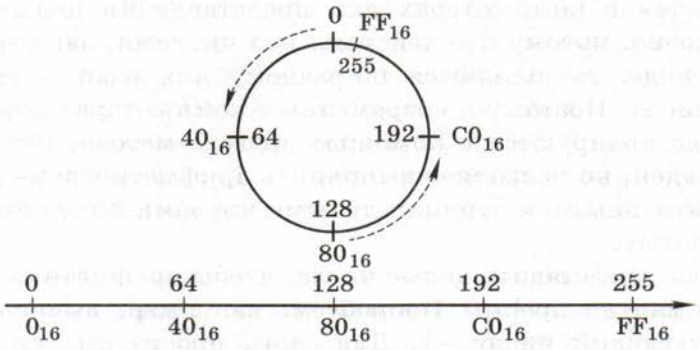


Рис. 4.6

Факт переполнения всегда фиксируется процессором, но выполнение программы не прерывается. Программе (точнее, программисту) предоставляется возможность как-то реагировать на переполнение или «не заметить» его.

Целые числа со знаком

Теперь рассмотрим числа со знаком (англ. *signed*). Для того чтобы различать положительные и отрицательные числа, в двоичном коде выделяется один бит для хранения знака числа — **знаковый разряд**. По традиции для этого используют самый старший бит, причём нулевое значение в нём соответствует знаку «плюс», а единичное — знаку «минус». Ноль формально является положительным числом, так как все его разряды, включая знаковый, нулевые.

Поскольку один бит выделяется для хранения информации о знаке, ровно половина из всех 2^K чисел будут отрицательными. Учитывая, что одно значение — нулевое, положительных чисел будет на единицу меньше, т. е. допустимый диапазон значений оказывается несимметричным.

Положительные числа записываются в знаковой форме так же, как и в беззнаковой, но для значения остаётся на один разряд меньше. А как поступить с отрицательными числами? Первое, что приходит в голову, это кодировать отрицательные значения точно так же, как и положительные, только записывать в старший бит единицу. Такой способ кодирования называется **прямым кодом**. Несмотря на свою простоту и наглядность, он не применяется в компьютерах для представления целых чисел[1]. Это неудобно, потому что действия над числами, записанными в прямом коде, выполняются по-разному для разных сочетаний знаков чисел. Поэтому в современных компьютерах отрицательные числа кодируются с помощью другого метода, который менее нагляден, но позволяет выполнять арифметические действия с положительными и отрицательными числами по одному и тому же алгоритму.

Как же представить целые числа, чтобы арифметика выглядела максимально просто? Попробуем, например, вычислить код, соответствующий числу –1. Для этого просто вычтем из нуля единицу:

$$\begin{array}{r|l} 1 & 0000\ 0000 \\ -\ 0 & 0000\ 0001 \\ \hline 0 & 1111\ 1111 \end{array}$$

Чтобы вычитание «состоялось», придется занять из несуществующего старшего бита единицу, что не очень естественно, но зато быстро приводит к правильному результату[2]. Заметим, что фактически мы вычитали не из 0, а из 256. В общем случае вычисление происходит по формуле $2^K - X$, где для данного примера $K = 8$, а $X = 1$.

---

[1] Тем не менее прямой код используется в представлении вещественных чисел.

[2] Для проверки можно прибавить к полученному коду единицу, в результате должен получиться ноль.

Однако предложенный способ перевода не слишком хорош, поскольку мы использовали дополнительный «несуществующий» разряд. Вместо этого можно использовать равносильный алгоритм:

$$256 - X = (255 - X) + 1 = \text{not } X + 1.$$

Здесь «not» обозначает логическую операцию «НЕ» (инверсию), применяемую к каждому биту числа отдельно (все нули заменяются на единицы и наоборот).

Итак, для получения кода целого числа $(-X)$ нужно:

Алгоритм А1

1) Выполнить инверсию каждого разряда двоичного представления числа X (такой код называется обратным).
2) К полученному результату прибавить единицу.

В результате получается **дополнительный код** — он *дополняет* число до 2^K.

Алгоритм А1 приводится в большинстве учебников, но его можно немного изменить так, чтобы облегчить человеку «ручные» вычисления:

Алгоритм А2

1) Вычислить число $X - 1$ и перевести его в двоичную систему.
2) Выполнить инверсию каждого разряда результата.

Оба алгоритма дают одинаковые результаты, но алгоритм А2 для человека существенно проще, потому что ему легче вычесть единицу в «родной» десятичной системе, чем прибавлять её в двоичной (при использовании алгоритма А1).

Наконец, оба пункта алгоритма А1 можно объединить, получив ещё один вариант:

Алгоритм А3

Выполнить инверсию всех старших битов числа, кроме последней (младшей) единицы и тех нулей, которые стоят после неё.

Например, определим дополнительный код числа «–16», которое хранится в 8-разрядной ячейке. Здесь $X = 16 = 10000_2$. Используя алгоритмы А1 и А2, получаем:

А1

X_2	$0001\ 0000_2$
not	$1110\ 1111_2$
+1	$1111\ 0000_2$

А2

$X-1$	15
$(X-1)_2$	$0000\ 1111_2$
not	$1111\ 0000_2$

Применение алгоритма А3 к числу $16 = 00010000_2$ сводится к замене первых трёх нулей единицами: $1111\ 0000_2$.

Для проверки можно сложить полученный результат с исходным числом и убедиться, что сумма обратится в ноль (перенос из старшего разряда не учитываем).

Повторное применение любого из алгоритмов А1–А3 всегда приводит к восстановлению первоначального числа (убедитесь в этом самостоятельно). Это свойство также удобно использовать для проверки.

В таблице 4.3 показаны шестнадцатеричные и двоичные коды некоторых характерных 8-разрядных чисел.

Таблица 4.3

X_{10}	–128	–127	...	–1	0	1	...	127
X_{16}	80	81	...	FF	00	01	...	7F
X_2	1000 0000	1000 0001	...	1111 1111	0000 0000	0000 0001	...	0111 1111

Обратите внимание на скачок при переходе от –1 к 0 и на два граничных значения: 127 и «–128». «Кольцо» для чисел со знаком выглядит так, как показано на рис. 4.7.

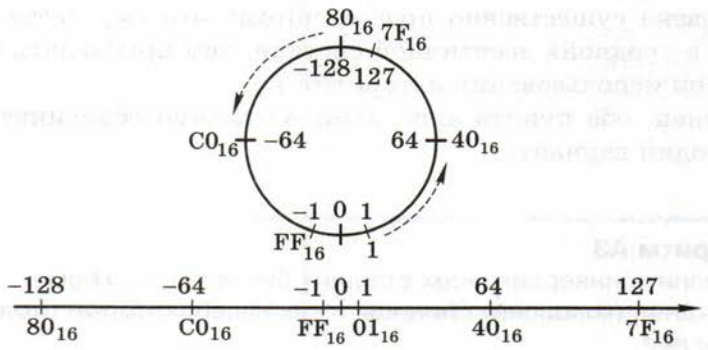


Рис. 4.7

Чтобы сравнить коды целых чисел без знака и со знаком, объединим обе таблицы (4.1 и 4.3) — получим табл. 4.4.

Таблица 4.4

Код	0	1	2	...	7F	80	81	...	FE	FF
Без знака	0	1	2	...	127	128	129	...	254	255
Со знаком	0	1	2	...	127	–128	–127	...	–2	–1

Общее количество значений со знаком и без знака одинаково, но их диапазоны сдвинуты друг относительно друга на числовой оси (рис. 4.8).



Рис. 4.8

В наших рассуждениях использовались 8-разрядные числа, но все выводы справедливы для чисел любой разрядности. От числа разрядов K зависят только граничные значения X_{max} и X_{min}, приведённые в табл. 4.5.

Таблица 4.5

K	8	16	32	64
X_{max}	127	32 767	2 147 483 647	9 223 372 036 854 775 807
X_{min}	–128	–32 768	–2 147 483 648	–9 223 372 036 854 775 808

Хотя дополнительный код гораздо менее нагляден, чем прямой, он значительно упрощает выполнение арифметических операций в компьютере. Например, вместо вычитания используется сложение с дополнительным кодом вычитаемого, поэтому не нужно проектировать специальное устройство для вычитания чисел.

Вопросы и задания

1. Чем отличается представление в компьютере целых чисел со знаком и без знака?
2. Приведите примеры величин, которые всегда имеют целые неотрицательные значения.
3. Как представлены в компьютере целые числа без знака?
4. Как изменится диапазон представления чисел, если увеличить количество разрядов на 1? На 2? На n?
5. Какое максимальное целое беззнаковое число можно записать с помощью K двоичных разрядов? Что произойдёт, если прибавить единицу к этому максимальному значению?
6. Как действует процессор при переполнении?
7. Почему максимальное положительное и минимальное отрицательное значения у целых двоичных чисел со знаком имеют разные абсолютные значения?
8. Верно ли, что положительные числа кодируются одинаково в знаковом и беззнаковом форматах?
9. Сформулируйте различные алгоритмы получения дополнительного кода для отрицательного числа.
10. Докажите, что алгоритмы А1, А2 и А3 всегда дают один и тот же результат.
11. Какое минимальное отрицательное значение можно записать с помощью K двоичных разрядов?
12. Может ли быть переполнение при сложении двух отрицательных чисел? Какой знак будет у результата?
13. Что получится, если правила перевода в дополнительный код применить к отрицательному числу?
14. Как можно проверить правильность перевода в дополнительный код?
15. В чём главное преимущество дополнительного кода при кодировании отрицательных чисел?
16. Почему компьютер может обойтись без вычитания?

Подготовьте сообщение

а) «Способы кодирования отрицательных целых чисел»

б) «Целочисленные типы данных в языках программирования»

Задачи

1. Цвет пикселя изображения кодируется как целое беззнаковое число. Найдите максимальное количество цветов при двух- и трёхбайтовом кодировании.
2. Используя арифметику 8-разрядных чисел без знака, выполните действия: 250 + 10 и 8 – 10. Объясните полученные результаты.
3. Выполните сложение десятичных чисел 65530 + 9 в 16-битной арифметике без знака.
4. Выполните сложение десятичных чисел 32760 + 9 в 16-битной арифметике со знаком.
5. Переведите в дополнительный код отрицательные числа –1, –10, –100 и запишите их с помощью 8 двоичных разрядов.
6. Постройте прямой код для отрицательных чисел –1, –10, –100, записанных с помощью 8 двоичных разрядов.
7. Рассматриваются 8-разрядные числа со знаком. Какие из приведённых шестнадцатеричных чисел отрицательные: 1, 8, F, 10, 18, 20, 30, 3F, 40, 70, 7F, 80, 90, A1, CC, F0, FF? Как это можно быстро определять?
8. Отвечая на вопрос учителя о том, как вычислить максимальное положительное и минимальное отрицательное значения у целых K-разрядных двоичных чисел со знаком, ученик ответил кратко: 2^{K-1}. В чём он ошибся, а в чём нет? Вычислите правильные значения для $K = 12$.
9. Каков будет результат операции 127 + 3 в 8-разрядной арифметике со знаком? Объясните полученный результат.
10. Факториалом называется произведение последовательных целых чисел, например 3! (читается «3 факториал») $= 1 \cdot 2 \cdot 3 = 6$. Вычисления выполняются в 16-разрядной целочисленной арифметике со знаком. Для какого максимального значения n удастся вычислить $n!$ и что получится при вычислении $(n+1)!$?

§ 28
Операции с целыми числами

Сложение и вычитание

Сложение и вычитание требуются не только для расчётов по формулам, но и для организации вычислений. Например, для того чтобы повторить какое-то действие R раз, используют переменную-счётчик, к которой после каждого выполнения этого действия прибавляют единицу, а затем результат сравнивают с R.

Вместо этого можно сразу записать в счётчик значение R и после каждого повторения вычитать из него единицу, пока не получится ноль[1].

Благодаря тому что отрицательные числа кодируются в дополнительном коде, при сложении можно не обращать внимания на знаки слагаемых, т. е. *со знаковым разрядом обращаются точно так же, как и со всеми остальными*.

Например, сложим числа 5_{10} ($0000\ 0101_2$) и -9_{10} ($1111\ 0111_2$), используя 8-разрядную двоичную арифметику. Применим сложение столбиком, не задумываясь о знаках чисел:

$$\begin{array}{r} 0000\ 0101 \\ +\ 1111\ 0111 \\ \hline 1111\ 1100 \end{array}$$

Для расшифровки получившегося отрицательного числа применим к нему схему получения дополнительного кода: $1111\ 1100 \to 0000\ 0100_2 = 4_{10}$. Таким образом, результат равен -4_{10}, что совпадает с правилами «обычной» арифметики.

При сложении двух чисел с одинаковыми знаками может случиться переполнение — сумма будет содержать слишком большое количество разрядов. Покажем, как это выглядит для положительных и отрицательных чисел.

Сложим десятичные числа 96 и 33. Их сумма 129 выходит за 8-битную сетку. Для того чтобы обнаружить переполнение, добавим к обоим слагаемым ещё один старший бит, совпадающий со знаковым (рис. 4.9).

$$\begin{array}{r} 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0 \\ +\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 1 \\ \hline 0\ 1\ 0\ 0\ 0\ 0\ 0\ 1 \\ S'\ S \end{array}$$

Рис. 4.9

Знаковый разряд S результата равен 1, т. е. сумма получилась отрицательной, хотя оба слагаемых положительны! Процессор определяет переполнение, сравнивая биты S и S': если они раз-

[1] Второй вариант более эффективен, потому что процессор автоматически сравнивает результат очередного действия с нулём.

личны, то произошло переполнение и результат неверный (см. рис. 4.9).

То же самое получается, если сложить два достаточно больших по модулю отрицательных числа, например –96 и –33. Добавим к кодам обоих чисел один старший разряд, равный знаковому разряду (рис. 4.10).

```
  1 | 1  0 1 0 0 0 0 0
+ 1 | 1  1 0 1 1 1 1 1
-----------------------
  1 | 0  1 1 1 1 1 1 1
  S'  S
```

Рис. 4.10

Получается, что в результате бит $S = 0$, хотя ответ должен быть отрицательным. Биты S' и S не совпадают, это говорит о том, что произошло переполнение. Несложно проверить (сделайте это самостоятельно), что, если переполнения нет, значения битов S и S' всегда одинаковы независимо от знаков слагаемых.

Сложение многоразрядных двоичных чисел в компьютере выполняет специальное устройство — **сумматор** (см. главу 3). Как мы уже говорили, вычитание сводится к сложению с дополнительным кодом вычитаемого, поэтому отдельного «блока вычитания» в компьютере нет.

Умножение и деление

Умножение и деление выполнять труднее, чем сложение и вычитание. Вспомните, например, что в математике умножение часто заменяют многократным сложением, а деление — многократным вычитанием.

К двоичным числам можно применять обычную схему умножения «столбиком». Перемножим, например, числа 9_{10} ($0000\ 1001_2$) и 5_{10} ($0000\ 0101_2$):

```
    00001001
  × 00000101
------------
    00001001
+ 00000000
  00001001
------------
  0000101101
```

Легко проверить, что это число равно 45_{10}.

По сравнению с десятичной системой, здесь есть серьёзное упрощение: первый сомножитель умножается на единицу (в этом случае результат равен ему самому) или на ноль (результат — 0). Поэтому компьютерное умножение целых чисел состоит из следующих элементарных действий:

1) вычисление очередного произведения в зависимости от младшего бита второго сомножителя: оно равно нулю (если этот бит нулевой) или первому сомножителю (если бит равен единице);
2) сложение содержимого сумматора с очередным произведением;
3) сдвиг содержимого первого сомножителя влево на 1 разряд;
4) сдвиг второго сомножителя вправо на 1 разряд (при этом следующий бит попадёт в младший разряд).

Таким образом, удаётся построить схему умножения без использования таблицы умножения. Заметим, что умножение — это довольно трудоёмкая операция, и для её ускорения конструкторы используют различные «хитрые» приёмы. Поэтому в реальных компьютерах всё может выглядеть значительно сложнее, чем в учебном примере.

Умножение, как и сложение, выполняется одинаково для положительных и отрицательных чисел (в дополнительном коде). Если в нашем примере вместо числа 9 подставить –9, то получится:

$$\begin{array}{r} 11110111 \\ \times\ 00000101 \\ \hline 11110111 \\ +\ 00000000 \\ 11110111 \\ \hline 100011010011 \end{array}$$

Оставив только 8 младших битов, можно убедиться (применяя алгоритмы А1–А3), что результат — это дополнительный код числа –45.

Теория деления нацело намного сложнее, чем приёмы умножения, поэтому мы её обсуждать не будем.

Сравнение

В отличие от арифметических действий операция сравнения *по-разному* выполняется для чисел со знаком и без него. Еще раз внимательно посмотрим на таблицы кодов 8-битных чисел, приведённые в § 27 (табл. 4.4). Если сравниваемые коды не превышают $7F_{16}$, то оба числа положительны и сравнение однозначно. Если это не так, то сравнение чисел с учётом и без учёта знака дает разные результаты. Например, для беззнаковых чисел 81_{16} (129_{10}) больше, чем $7F_{16}$ (127_{10}). Для чисел со знаком, наоборот, отрицательное значение 81_{16} (-127_{10}) будет меньше, чем $7F_{16}$ (127_{10}). Поэтому современные процессоры имеют разные команды для сравнения чисел со знаком и без знака. Чтобы не путаться, при сравнении с учётом знака обычно используют термины «больше»/«меньше», а при сравнении без учёта знака — «выше»/«ниже».

Поразрядные логические операции

В главе 3, изучая основы математической логики, мы увидели, что обработка истинности и ложности высказываний может быть представлена как набор операций с двоичными кодами. Оказывается, что логические операции, введённые первоначально для обработки логических данных, можно формально применить к битам двоичного числа, и такой подход широко используется в современных компьютерах.

Рассмотрим электронное устройство для управления гирляндой лампочек. Состояние каждой лампочки будет задаваться отдельным битом в некотором управляющем регистре: если бит равен нулю, лампочка выключена, если единице — включена. Для получения различных световых эффектов (типа «бегущих огней») требуется зажигать или гасить отдельные лампочки, менять их состояние на противоположное и т. д. (рис. 4.11). Точно так же биты регистров используются для управления внешними устройствами.

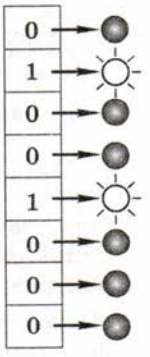


Рис. 4.11

Будем применять логические операции к каждому биту числа, как обычно, считая, что 1 соответствует значению «истина», а 0 — «ложь». Эти операции часто называют **поразрядными** или **битовыми**,

поскольку действия совершаются над каждым разрядом в отдельности, независимо друг от друга[1].

Введём несколько терминов, которые используются в литературе по вычислительной технике. **Сброс** — это запись в бит нулевого значения, а **установка** — запись единицы. Таким образом, если бит в результате какой-то операции становится равным нулю, то говорят, что он сбрасывается. Аналогично, когда в него записана единица, говорят, что бит установлен.

Маска — это константа (постоянная), которая определяет область применения логической операции к битам многоразрядного числа. С помощью маски можно скрывать (защищать) или открывать для выполнения операции отдельные биты[2].

Основные логические операции в современных процессорах — это «НЕ» (not), «И» (and), «ИЛИ» (or) и «исключающее ИЛИ» (xor).

Логическое «НЕ» (инвертирование, инверсия, not) — это замена всех битов числа на обратные значения: 0 на 1, а 1 — на 0. Эта операция используется, например, для получения дополнительного кода отрицательных чисел (см. алгоритм А1 в § 27). «НЕ» — это *унарная операция*, т. е. она действует на все биты *одного* числа. Маска здесь не используется.

D	*M*	*D* and *M*
0	0	0
1	0	0
0	1	0
1	1	1

Рис. 4.12

Логическое «И» (and). Обозначим через D содержимое некоторого бита данных, а через M — значение соответствующего ему бита маски. Операция «И» между ними задаётся таблицей, показанной на рис. 4.12.

Из таблицы видно, что при выполнении логического «И» нулевой бит в маске всегда сбрасывает (делает равным нулю) соответствующий бит результата, а еди-

[1] Для сравнения, сложение (как и другие арифметические действия) не является поразрядной операцией, поскольку возможен перенос из младшего разряда в старший.

[2] Использование маски аналогично выделению области рисунка в графическом редакторе — для выделенных пикселей маска равна 1, для остальных — нулю.

ничный бит позволяет сохранить значение D (как бы пропускает его, открывая «окошко»).

С помощью логической операции «И» можно *сбросить* отдельные биты числа (те, для которых маска нулевая), не меняя значения остальных битов (для которых в маске стоят единицы).

Например, операция X and 1 сбросит у любого числа X все биты, кроме самого младшего. С помощью этого приёма легко узнать, является ли число чётным: остаток от деления на 2 равен последнему биту!

Логическое «ИЛИ». Вспомнив таблицу истинности логической операции «ИЛИ» (or), можно обнаружить, что в ноль в маске сохраняет бит (X or 0 = X), а единица устанавливает соответствующий бит результата (X or 1 = 1) (рис. 4.13).

D	M	D or M
0	0	0
1	0	1
0	1	1
1	1	1

Рис. 4.13

С помощью логической операции «ИЛИ» можно *установить* отдельные биты числа (те, для которых маска единичная), не меняя значения остальных битов (для которых в маске стоят нули).

Например, операция X or 80_{16} установит старший бит восьмиразрядного числа X, формально сделав тем самым число отрицательным.

Таким образом, используя операции «И» и «ИЛИ», можно сбрасывать и устанавливать любые биты числа, т. е. *строить любой нужный двоичный код*. Где это может пригодиться? Рассмотрим примеры решения конкретных задач.

Пример 1. На клавиатуре набраны 3 цифры, образующие значение целого числа без знака. Определить, какое число было введено.

При нажатии клавиши на клавиатуре в компьютер поступает код нажатой клавиши. Выпишем десятичные и шестнадцатеричные коды всех символов, обозначающих цифры:

Символ	'0'	'1'	'2'	'3'	'4'	'5'	'6'	'7'	'8'	'9'
X_{10}	48	49	50	51	52	53	54	55	56	57
X_{16}	30	31	32	33	34	35	36	37	38	39

Будем пользоваться шестнадцатеричными кодами: как видно из таблицы, их связь с цифрами числа гораздо нагляднее. Чтобы получить числовое значение цифры из кода символа X, достаточно сбросить его старшие четыре бита, не изменяя значений четырёх младших битов. Для этого нужно использовать операцию X and $0F_{16}$.

Пусть S_1 — код первого введённого символа, S_2 — второго, S_3 — третьего, а N обозначает искомое число. Тогда алгоритм перевода кодов символов в число выглядит так:

1. $N = 0$.
2. $W = S_1$ and $0F_{16}$ (выделяем первую цифру).
3. $N = 10 \cdot N + W$ (добавляем её к числу).
4. $W = S_2$ and $0F_{16}$ (выделяем вторую цифру).
5. $N = 10 \cdot N + W$ (добавляем её к числу).
6. $W = S_3$ and $0F_{16}$ (выделяем третью цифру).
7. $N = 10 \cdot N + W$ (добавляем её к числу).

Пусть, например, набраны символы '1', '2' и '3'. Тогда по таблице находим, что $S_1 = 31_{16}$, $S_2 = 32_{16}$ и $S_3 = 33_{16}$. Значение W на втором шаге вычисляется так:

$$\begin{array}{rr} & 0011\ \ 0001 \\ \text{and} & 0000\ \ 1111 \\ \hline & 0000\ \ 0001 \end{array}$$

Так как $W = 1$, на третьем шаге получаем $N = 1$. Следующая пара шагов — четвёртый и пятый — дают результаты $W = 2$ и $N = 12$ соответственно. Наконец, результат завершающих шагов: $W = 3$ и $N = 123$.

Такая процедура используется в каждом компьютере: именно так коды цифровых символов, набранные на клавиатуре, преобразуются в числа, с которыми компьютер выполняет арифметичес-

кие действия. Заметим, что фактически здесь использована схема Горнера для представления целого числа (см. § 10).

Пример 2. Создадим структуру данных S, которая отражает, есть или нет в некотором числе каждая из цифр от 0 до 9. В математике такая структура называется **множеством**. Для хранения S будем использовать 16-разрядное целое число (рис. 4.14).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						1	0	0	1	0	0	1	0	0	1

Рис. 4.14

Договоримся, что младший бит числа имеет номер 0 и хранит информацию о том, есть во множестве цифра 0 (если этот бит равен 0, такой цифры нет, если равен 1, то есть). Аналогично первый бит (второй по счёту справа) показывает, есть ли во множестве цифра 1, и т. д. Старшие биты 10–15 при этом не используются. Например, во множестве, изображенном на рис. 4.14, есть только цифры 0, 3, 6 и 9.

Для записи элементов во множество и проверки их наличия удобно использовать логические операции. Рассмотрим для примера бит 5. Маска, которая потребуется для обращения к нему, — это единица в пятом разряде и нули во всех остальных, т. е. $M = 0020_{16}$. С её помощью можно добавить элемент к множеству с помощью операции «ИЛИ»: $S = S$ or M. А узнать, есть ли во множестве интересующая нас цифра, можно, выделив соответствующий бит с помощью логического «И» ($P = S$ and M) и проверив результат на равенство нулю.

Исключающее ИЛИ. Как видно из таблицы истинности, операция «исключающее ИЛИ» (xor) не изменяет биты, когда маска нулевая, и меняет биты на противоположные при единичной маске (рис. 4.15).

Например, команда $Y = Y$ xor FF_{16} выполняет инверсию всех битов 8-разрядного целого числа. Напомним, что это один из этапов получения дополнительного кода отрицательных чисел.

D	M	D xor M
0	0	0
1	0	1
0	1	1
1	1	1

Рис. 4.15



С помощью логической операции «исключающее ИЛИ» можно выполнить *инверсию* отдельных битов числа (тех, для которых маска единичная), не меняя значения остальных битов (для которых в маске стоят нули).

Пример 3. Пусть X — это результат выполнения некоторого вычислительного теста, а Y — то, что ожидалось получить («правильное» значение). Нужно определить, в каких разрядах различаются эти числа (для инженера это очень полезная подсказка, где искать неисправность).

Предположим, что $X = 7$, $Y = 3$. В результате операции X xor Y устанавливаются (в единицу) только те разряды, которые в этих числах не совпали, а остальные сбрасываются[1]. В данном случае находим, что числа различаются только одним битом:

$$\begin{array}{rr} & 0000\ \ 0111 \\ \text{xor} & 0000\ \ 0011 \\ \hline & 0000\ \ 0100 \end{array}$$

Пример 4. Используя логическую операцию «исключающее ИЛИ», можно шифровать любые данные. Покажем это на примере простого текста '2*2=4'.

Выберем любую маску, например $23_{10} = 0001\ 0111_2$. Эта маска представляет собой *ключ шифра* — зная ключ, можно расшифровать сообщение. Возьмём первый символ — цифру '2', которая имеет код $50_{10} = 0011\ 0010_2$, и применим операцию «исключающее ИЛИ» с выбранной маской:

$$\begin{array}{rr} & 0011\ \ 0010 \\ \text{xor} & 0001\ \ 0111 \\ \hline & 0010\ \ 0101 \end{array}$$

Полученное значение $0010\ 0101_2 = 37_{10}$ — это код символа '%'. Для расшифровки применим к этому коду «исключающее ИЛИ» с той же маской:

$$\begin{array}{rr} & 0010\ \ 0101 \\ \text{xor} & 0001\ \ 0111 \\ \hline & 0011\ \ 0010 \end{array}$$

[1] Профессиональные программисты часто используют операцию xor для обнуления переменной: команда `R := R xor R` запишет в переменную R ноль, независимо от её начального значения.

В результате получили число 50 — код исходной цифры '2'.

Повторное применение операции «исключающее ИЛИ» с той же маской *восстанавливает исходное значение*, т. е. эта логическая операция *обратима*.

Если применить такую процедуру шифрования ко всем символам текста '2*2=4', то получится зашифрованный текст '%=%*#'.

Обратимость операции «исключающее ИЛИ» часто используется в компьютерной графике для временного наложения одного изображения на другое. Это может потребоваться, например, для выделения области с помощью инвертирования её цвета.

Сдвиги

Об операции сдвига вспоминают гораздо реже, чем она того заслуживает. Перечитайте ещё раз алгоритм умножения, описанный выше, и вы убедитесь, что он весь построен на сдвигах. Сдвиги незаменимы тогда, когда требуется проделать ту или иную обработку *каждого* бита, входящего в число. Наконец, сдвиги двоичного числа позволяют быстро умножить или разделить число на степени двойки: 2, 4, 8 и т. д. Поэтому программисты очень ценят и широко применяют всевозможные разновидности сдвигов.

Идея операции сдвига довольно проста: все биты кода одновременно сдвигаются в соседние разряды[1] влево или вправо (рис. 4.16).

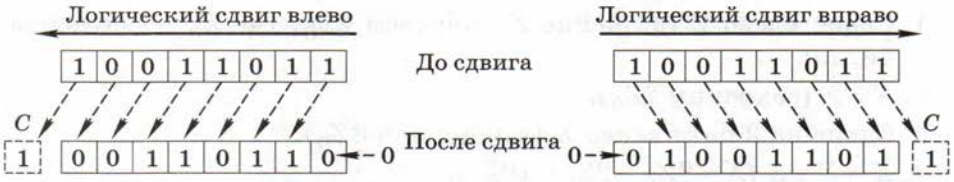


Рис. 4.16

[1] Аппаратно сдвиг реализуется необычайно просто и изящно: регистр, содержащий число, сбрасывается в ноль, при этом из тех разрядов, где исчезла единица, электрический импульс проходит в соседние и устанавливает их в единицу. При этом важно, что все разряды обрабатываются *одновременно*.

Отдельно надо поговорить о двух крайних битах, у которых «нет соседей». Для определённости обсудим сдвиг влево. Для самого младшего бита (на рис. 4.16 он крайний справа) данные взять неоткуда, поэтому в него просто заносится ноль. Самый старший (крайний слева) бит должен потеряться, так как его некуда сохранить. Чтобы данные не пропали, содержимое этого разряда копируется в специальную ячейку процессора — **бит переноса**[1] *C* (от англ. *carry* — перенос), с которым может работать процессор.

Рассмотренный тип сдвига обычно называется **логическим сдвигом**. Его можно использовать для быстрого умножения и деления. Рассмотрим, например, 8-разрядный двоичный код 0000 1100, который представляет число 12_{10}. Выполнив логический сдвиг влево, получим 0001 1000, т. е. число 24_{10}, которое вдвое больше! Это не случайность: вспомните, что происходит, если к десятичному числу справа приписать дополнительный ноль, например 34 → 340.

При сдвиге вправо любое чётное число уменьшается ровно в 2 раза. В случае нечётного значения происходит деление нацело, при котором остаток отбрасывается. Например, из 0001 0001 = 17_{10} при сдвиге вправо получается 0000 1000 = 8_{10}.



Логический сдвиг влево на 1 разряд увеличивает целое положительное число вдвое, а сдвиг вправо делит на 2 нацело.

Пример. Для умножения числа, находящегося в ячейке *Z*, на 10 можно использовать такой алгоритм:

1. Сдвиг влево *Z* (в ячейке *Z* получаем $2Z_0$, где Z_0 — исходное число).
2. $X = Z$ (сохраним $2Z_0$).
3. Сдвиг на 2 бита влево *X* (вычислили $8Z_0$).
4. $X = X + Z$ ($X = 8Z_0 + 2Z_0 = 10Z_0$).

Для некоторых компьютеров такая последовательность выполняется быстрее, чем стандартная операция умножения.

Посмотрим, что получится для отрицательных чисел. Сдвинем влево код 1111 1000 (8-разрядное представление числа –8):

[1] При программировании на языках высокого уровня бит переноса недоступен.

получится 1111 0000. Легко проверить, что это дополнительный код числа –16, т. е. значение удвоилось! Но со сдвигом вправо ничего не получается: из 1111 1000 получаем 0111 1100 — это код положительного числа! Дело в том, что при сдвиге вправо отрицательных чисел, в отличие от положительных, старший разряд надо заполнять не нулём, а единицей! Чтобы исправить положение, вводится ещё одна разновидность сдвига — **арифметический сдвиг**. Его единственное отличие от логического состоит в том, что старший (знаковый) бит не меняется, т. е. знак числа остаётся прежним (рис. 4.17).



Рис. 4.17

Если применить арифметический сдвиг к коду 1111 1000, получается 1111 1100 — дополнительный код числа –4, т. е. произошло деление на 2. В качестве упражнения проверьте, как ведёт себя отрицательное нечётное число при арифметическом сдвиге вправо.

Арифметический сдвиг влево не требуется, поскольку он ничем не отличается от обычного логического сдвига.

То, что в результате логических сдвигов содержимое крайних разрядов теряется, не всегда удобно. Поэтому в компьютере предусмотрен **циклический сдвиг**, при котором бит из одного крайнего разряда переносится в другой («по циклу», рис. 4.18).



Рис. 4.18

Циклический сдвиг позволяет «просмотреть» все биты и вернуться к исходному значению. Если сделать последовательно 8 циклических сдвигов 8-битного числа, каждый его бит на каком-то шаге окажется на месте младшего разряда, где его можно выделить с помощью логической операции «И» с маской 1. Так можно «просматривать» не только младший, но и любой другой разряд (например, для выделения старшего разряда нужно использовать маску 80_{16}).

Вопросы и задания

1. Покажите на примере, как складываются два положительных целых числа, записанные в 8-разрядные ячейки. Что изменится, если числа будут отрицательными?
2. Что такое дополнительный код? Сформулируйте правила получения дополнительного кода числа.
3. При каких комбинациях знаков слагаемых в результате сложения может возникнуть переполнение?
4. Какое устройство выполняет в компьютере сложение? Вспомните, что вы знаете об этом устройстве.
5. Почему не нужно разрабатывать специальное устройство для вычитания целых чисел?
6. Перемножьте столбиком два положительных целых числа в двоичной системе счисления. Изменится ли алгоритм выполнения операции, если у одного из сомножителей поменять знак?
7. Почему коды чисел со знаком и без знака нужно сравнивать по-разному?
8. Что такое поразрядные операции? Приведите примеры.
9. Почему арифметические операции нельзя отнести к поразрядным?
10. Что такое маска?
11. Как, используя маску, сбросить определённый бит (записать в него 0)?
12. Напишите значение маски для того, чтобы сбросить в 16-разрядном числе 2 младших бита, не изменяя все остальные. Какую логическую операцию нужно для этого использовать?
13. Как, используя маску, установить определённый бит?
14. Напишите значение маски для того, чтобы установить в 16-разрядном числе 2 старших бита, не изменяя все остальные. Какую логическую операцию нужно для этого использовать?
15. Как, используя логические операции, определить, делится ли число на 4? На 8?
16. В каких практических задачах можно применять установку или сброс битов двоичного кода?
17. Каковы возможности операции «исключающее ИЛИ»?

*18. Попробуйте придумать алгоритм шифрования кода с помощью операции «исключающее ИЛИ». Постарайтесь предложить простой алгоритм изменения маски, а не просто использовать константу.
19. Прочитайте ещё раз материал, связанный с переполнением при сложении. Какой логической операцией можно определить, совпадают или нет биты S' и S?
20. Какую роль играет операция «НЕ» при получении отрицательных чисел?
21. Как выполнить инверсию всех битов, не используя логическую операцию «НЕ»?
22. Что такое сдвиг? Какие вы знаете виды сдвига?
23. Как обрабатываются самый старший и самый младший биты при различных типах сдвига?
24. Покажите на примерах, что сдвиг влево двоичного кода удваивает число, а сдвиг вправо — уменьшает вдвое.
25. Почему логический сдвиг не годится для уменьшения в два раза отрицательных чисел? Как работает арифметический сдвиг?
26. Почему не требуется арифметический сдвиг влево?
*27. Выведите правило вычисления результата арифметического сдвига отрицательного нечётного числа на один разряд вправо. Проверьте, применимо ли это правило к положительным нечётным числам. Как упрощается формула для чётных исходных значений?
28. Где могут применяться сдвиги?

Подготовьте сообщение

а) «Битовые логические операции»
б) «Шифрование с помощью операции "исключающее ИЛИ"»
в) «Применение сдвигов»

Задачи

1. Переведите в 8-разрядный двоичный код десятичные числа 31 и 19 и сложите их. Для проверки переведите полученную сумму в десятичную систему счисления.
2. Повторите вычисления предыдущей задачи, заменив первое слагаемое на –31. Подумайте, что изменится, если код сделать 16-разрядным?
3. Выберите произвольные значения двух целых чисел A и B и запишите их в виде 8-разрядных двоичных кодов. Проверьте путем непосредственных вычислений справедливость тождества $A - B = A + (-B)$.

4. Сложение ведётся в 8-разрядной арифметике со знаком. Какое максимальное число можно прибавить к двоичной константе 100000_2, чтобы не возникло переполнения? Как изменится результат, если число будет беззнаковым?
5. Переведите в двоичный код десятичные числа 12 и 7 и перемножьте их. Для проверки переведите результат в десятичную систему счисления.
6. Повторите вычисления предыдущей задачи, заменив первый сомножитель на −12. Считайте, что числа представлены в 8-разрядном коде.
7. Братья Петя и Коля часто спорят по поводу решения задач по информатике. Главная причина состоит в том, что Петя всегда решает задачу, как показал учитель, а Коля вечно придумывает что-то своё, причём не всегда удачно. Сегодня, например, они осваивали двоичную арифметику, умножая 1000_2 на 11011_2. Петя добросовестно умножал столбиком, а Коля взял второй сомножитель и, приписав к нему три нуля, получил такой же ответ. После объяснений Петя был вынужден признать правоту брата. Как объяснил свое решение Коля?
8. Какое из двух беззнаковых чисел больше: 0111 0111 или 1000 1000? Изменится ли ваш ответ, если вам скажут, что исходные коды — это 8-разрядные числа со знаком? Переведите оба значения для случаев чисел со знаком и без него в десятичную систему счисления.
9. Код строчной латинской буквы 'a' равен 61_{16}, а заглавной 'A' — 41_{16}. Используя логическую операцию «И», преобразуйте код строчной буквы в код заглавной. Проверьте, работает ли предложенный вами метод для других букв.
10. Используя логическую операцию «ИЛИ», преобразуйте код заглавной буквы 'A' в код строчной 'a'. Проверьте, работает ли предложенный вами метод для других букв.
11. Петя и Коля решают домашнюю задачу: известны коды двух введённых цифр C_1 и C_2. Найти сумму этих цифр. Петя, как обычно, глядя на решение задач в классе, пишет:
    1) $N_1 = C_1$ and 30_{16};
    2) $N_2 = C_2$ and 30_{16};
    3) $S = N_1 + N_2$.

    Коля предлагает более короткое решение:
    1) $S = C_1 + C_2$;
    2) $S = S$ and $0F_{16}$.

    Петя, ссылаясь на образцы задач в учебнике, критикует маску $0F_{16}$. Но Коля показывает на двух примерах ('2' и '3'; '5' и '7'), что его алгоритм даёт правильные результаты. Что скажет учитель по поводу Колиного решения?

12. Выполните битовую операцию X and 3 для следующих десятичных значений X: 4, 5, 8, 15, 16. Для каких из них получился нулевой ответ? Что общего у этих чисел?

13. Разработайте аналогичные способы определения делимости на 2, 8 и 16.

***14.** Попробуйте разработать алгоритм, который позволяет поменять местами значения двух ячеек памяти, используя только операцию «исключающее ИЛИ».

15. Цвет точки в формате RGB хранится как 4-байтовое целое число, которое в шестнадцатеричном виде выглядит так: 00 RR GG BB (т. е. старший байт не используется, а в каждом из последующих байтов хранится одна из трёх цветовых компонент[1]). Напишите последовательность операций, позволяющих выделить из 32-битного числа каждую из трёх цветовых компонент. Какая из них потребует большего числа операций?

16. Петя и Коля решают задачу: цвет точки в формате RGB хранится как 4-байтовое целое число N, которое в шестнадцатеричном виде выглядит так: 00 RR GG BB. Написать последовательность операций, позволяющих выделить из 32-битного числа красную компоненту. Петино решение:
1) $N = N$ and $\mathrm{FF0000}_{16}$;
2) логический сдвиг N вправо на 16 разрядов.
Колино решение содержит только вторую из этих операций. Чьё решение правильное?

17. Используя только сдвиги, сбросьте 4 старших разряда 8-битного значения. Как с помощью сдвигов сбросить 4 младших разряда?

18. Каков результат логического сдвига влево на 4 разряда шестнадцатеричного целого числа FEDC_{16}? Сравните его с результатом циклического сдвига.

19. Заданы два шестнадцатеричных целых числа: 1234_{16} и FEDC_{16}. К каждому из них применяются логический, циклический и арифметический сдвиги вправо на 4 разряда (каждый раз сдвигается первоначальное значение, а не результат предыдущего сдвига!) Напишите и объясните результаты для каждой операции.

---

[1] Несмотря на то что старший байт кажется лишним, этот способ хранения не лишён смысла. Дело в том, что процессор не приспособлен к обработке 3-байтовых данных, тогда как с 4-байтовыми работает очень быстро. Описанный формат, в частности, применяется при хранении таблиц цветов в графическом формате BMP. В других форматах (например, в PNG) старший байт используется для хранения степени прозрачности пикселя (альфа-канала).

20. Запишите число –18 в 8-разрядном двоичном коде. Что получится, если применить к нему логический сдвиг вправо? Арифметический сдвиг вправо? Сравните полученные результаты и объясните их.

21. Переведите число –1 в дополнительный двоичный код и дважды примените к нему арифметический сдвиг вправо. Какой будет результат?

22. Выполните приведённый в тексте параграфа алгоритм умножения на 10 для $Z = 1100_2$. Для проверки переведите результат в десятичную систему счисления.

§ 29
Хранение в памяти вещественных чисел

В начале главы мы отмечали принципиальное различие между вещественными и целыми числами: целые числа дискретны, а вещественные, напротив, непрерывны, а значит, не могут быть полностью корректно перенесены в дискретную по своей природе вычислительную машину. Как же всё-таки кодируются в компьютерах вещественные числа?

В первых ЭВМ использовалось кодирование **с фиксированной запятой**. Это значит, что положение запятой, отделяющей целую часть от дробной, было жёстко закреплено в разрядной сетке конкретной ЭВМ — раз и навсегда для всех чисел и для всех технических устройств этой машины. Все вычислительные алгоритмы были заранее «настроены» на это фиксированное размещение. Но в задачах, которые решаются на компьютерах, встречаются самые разнообразные по величине числа, от размера атома до астрономических расстояний. Чтобы согласовать их с таким жёстким представлением, программист, подготавливая задачу к решению на ЭВМ, выполнял большую предварительную работу по *масштабированию* данных: маленькие числа умножались на определённые коэффициенты, а большие, напротив, делились. Масштабы подбирались так, чтобы результаты всех операций, включая промежуточные, не выходили за пределы разрядной сетки и в то же время обеспечивалась максимально возможная точность (все разряды данных по возможности находились в пределах сетки). Эта работа требовала много времени и часто являлась источником ошибок.

Тем не менее работа с фиксированным размещением запятой не только показала *недостатки* метода, но и наметила путь их устранения. В самом деле, если наиболее сложным и трудоёмким местом является масштабирование данных, надо его автоматизировать. Иными словами, надо научить машину самостоятельно размещать запятую так, чтобы числа при счёте не выходили за разрядную сетку и по возможности сохранялись с максимальной точностью. Конечно, для этого нужно разрешить компьютеру «перемещать» запятую, а значит, дополнительно как-то сохранять в двоичном коде числа информацию о ее текущем положении. В этом и заключается главная идея представления чисел **с плавающей запятой**[1].

Удобное представление вещественных чисел не пришлось специально придумывать. В математике уже существовал подходящий способ записи, основанный на том, что любое число A в системе счисления с основанием B можно записать в виде

$$A = \pm Z \cdot B^P,$$

где Z называют **значащей частью**, а показатель степени P — **порядком** числа (рис. 4.19). Для десятичной системы это выглядит привычно, например, заряд электрона равен $-1{,}6 \cdot 10^{-19}$ кулона, а скорость света в вакууме составляет $3 \cdot 10^8$ м/с.

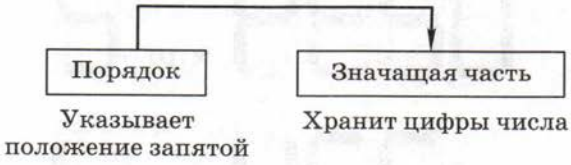


Рис. 4.19

Однако представление числа с плавающей запятой не единственно. Например, число 23,4 можно записать следующими способами:

$$2340 \cdot 10^{-2} = 234 \cdot 10^{-1} = 23{,}4 \cdot 10^0 = 2{,}34 \cdot 10^1 = \\ = 0{,}234 \cdot 10^2 = 0{,}0234 \cdot 10^3 = \ldots$$

На первый взгляд, выбор очень широкий, однако большинство вариантов обладают серьезными недостатками. В частности,

[1] В англоязычных странах используется термин *floating point — плавающая точка*, поскольку в этих странах традиционно целая часть отделяется от дробной не запятой, как у нас, а точкой.

все представления, в которых значащая часть содержит нули непосредственно после запятой (0,0234, 0,00234 и т. п.) или перед ней (2340, 23400 и т. п.), не подходят, поскольку, сохраняя эти незначащие нули, мы напрасно увеличиваем разрядность чисел. Согласно математической теории, для обеспечения максимальной точности при сохранении цифр числа в фиксированном количестве разрядов надо выбирать такой метод, при котором значащие цифры числа следует поместить как можно ближе к запятой. С этой точки зрения оптимальным будет вариант, когда целая часть равна нулю, а первая ненулевая цифра находится сразу после запятой (в нашем примере 0,234). При этом вместо двух частей (целой и дробной) остаётся только дробная, что фактически делает ненужной «разделительную» запятую.

Но взгляните на рис. 4.20, *а*, изображающий такое число на индикаторе: первый разряд всегда равен нулю, что делает его практически бесполезным. Поэтому с точки зрения экономии разрядов лучше взять другой вариант, в котором значащая часть равна 2,34 (рис. 4.20, *б*). Именно такой выбор закреплён в стандарте IEEE 754[1], на котором основана арифметика вещественных чисел в современных компьютерах.

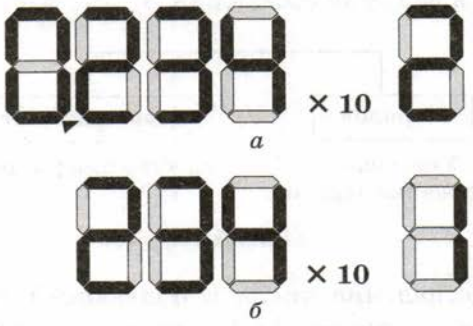


Рис. 4.20

Итак, существуют два приблизительно равноценных способа представления чисел с плавающей запятой:

- оптимальный с теоретической точки зрения, в котором целая часть нулевая, а первая цифра дробной части ненулевая ($0{,}234 \cdot 10^2$);

[1] Последняя версия стандарта называется IEEE 754-2008.

- более удобный с практической точки зрения, в котором целая часть состоит из единственной ненулевой цифры ($2{,}34 \cdot 10^1$).

К сожалению, эта «двойственность» порождает некоторую путаницу. В теоретической литературе, как правило, используется первый способ[1]. Все описания конкретных компьютерных систем, напротив, базируются на втором. Причём в обоих случаях обычно используется один и тот же русский термин — **мантисса**. Зато в англоязычной компьютерной литературе приняты два разных термина: в первом случае значащая часть называется *mantissa* (слово «мантисса» для математиков однозначно связано с дробной частью числа), а во втором — *significand* (значащая часть).

Далее мы будем использовать второй вариант, поскольку именно он даёт возможность решать задачи, связанные с практическим кодированием вещественных чисел. В связи с этим мы будем применять термин «значащая часть», а не «мантисса».

В компьютере используется такое представление вещественных чисел с плавающей запятой, при котором значащая часть Z удовлетворяет условию $1 \le Z < B$, где B — основание системы счисления. Такое представление называется **нормализованным**.

Нормализованное представление числа единственно — в нашем примере это $2{,}34 \cdot 10^1$. Любое число может быть легко нормализовано. Единственное, но важное исключение из правила составляет нуль — для него невозможно получить $Z \ge 1$. Ради такого важного случая было введено дополнительное соглашение: число 0, в котором все биты нулевые, в качестве исключения считается нормализованным.

Всё сказанное выше можно применить и к двоичной системе:

$$A = \pm Z \cdot 2^P, \text{ причём } 1 \le Z < 2.$$

Например: $-7_{10} = -111 \cdot 2^0 = -1{,}11 \cdot 2^{10}$ (не забывайте, что значащая часть и порядок записаны в двоичной системе!); отсюда $Z = 1{,}11$ и $P = 10$. Двоичная значащая часть *всегда* (исключая, разумеется, ноль!) *начинается с единицы*, так как $1 \le Z < 2$. Поэтому

[1] К этой группе относится большинство отечественных книг по основам вычислительной техники.

во многих компьютерах (в том числе в компьютерах на базе процессоров Intel) эта так называемая **скрытая единица** не хранится в ОЗУ, что позволяет сэкономить еще один дополнительный разряд значащей части[1].

Идея «скрытой единицы» раньше действительно давала заметное увеличение точности представления чисел, поскольку количество разрядов в устройствах ЭВМ того времени было невелико и поэтому усложнение метода кодирования было оправдано. Сейчас, когда процессоры работают с 64-битными данными, это скорее дань традиции, чем практически полезная мера[2].

Таким образом, при кодировании вещественного числа с плавающей запятой фактически *хранятся две величины: его значащая часть* (significand) *и порядок*. От разрядности значащей части зависит точность вычислений, а от разрядности порядка — диапазон представления чисел. В таблице 4.6 приведены характеристики стандартных вещественных типов данных, используемых в математическом сопроцессоре Intel.

Таблица 4.6

Тип	Диапазон	Число десятичных значащих цифр	Размер (байтов)
single	$1{,}4 \cdot 10^{-45} - 3{,}4 \cdot 10^{38}$	7–8	4
double	$4{,}9 \cdot 10^{-324} - 1{,}8 \cdot 10^{308}$	15–16	8
extended	$3{,}6 \cdot 10^{-4951} - 1{,}2 \cdot 10^{4932}$	19–20	10

Рассмотрим, как «распланированы» 4 байта, отводимые под простейший тип *single*. Тип *double* устроен совершенно аналогично, а тип *extended*, который является основным форматом для

---

[1] В результате то, что осталось после «скрытия» единичной целой части, можно вполне обоснованно называть мантиссой.

[2] Оценим величину добавки для математического сопроцессора Intel. Значащая часть чисел двойной точности вместо «скрытой единицы» приобретает дополнительный 53-й (!) бит, что прибавляет к значению числа поправку $2^{-53} \approx 1{,}1 \cdot 10^{-16}$, влияющую на 16-й десятичный знак; согласно IEEE 754-2008, в 128-битовых числах эта поправка будет и того меньше: $2^{-113} \approx 9{,}6 \cdot 10^{-35}$.

вычислений в математическом сопроцессоре, отличается только тем, что в нём единица в целой части не «скрывается».

В числах типа *single* 23 младших бита (с номерами от 0 до 22) хранят значащую часть числа, следующие 8 битов (с 23 по 30) — порядок, а старший (31-й) бит отведен под знак числа (рис. 4.21).

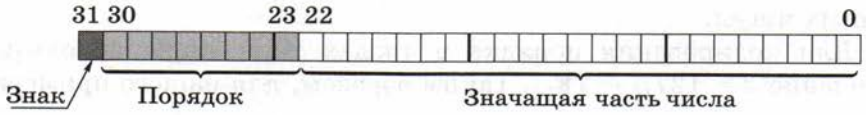


Рис. 4.21

Правила двоичного кодирования вещественных чисел во многом отличаются от правил кодирования целых чисел. Для того чтобы в них разобраться, рассмотрим конкретный пример — закодируем число –17,25 в формате *single*. Прежде всего, переведём модуль числа в двоичную систему, отдельно целую и дробную части (см. § 11):

$$17{,}25 = 10001{,}01_2.$$

Для нормализации нужно передвинуть запятую на $4_{10} = 100_2$ разряда влево:

$$10001{,}01 \cdot 2^0 = 1{,}000101 \cdot 2^{100}.$$

Построим значащую часть, «скрыв» единицу в целой части:

$$M = Z - 1 = 0{,}0001010...0.$$

Так как число отрицательное, знаковый разряд нужно установить в 1, т. е. $S = 1$.

В отличие от целых чисел значащая часть вещественных чисел хранится в прямом коде.

Таким образом, значащие части положительного и равного по модулю отрицательного числа одинаковы, а отличаются они только старшим (знаковым) битом.

Теперь остаётся закодировать двоичный порядок 100. Порядок — это целое число со знаком, для него используется **кодиро-**

вание со смещением: чтобы вообще избавиться от знака порядка, к нему добавляют некоторое положительное смещение d:

$$P_d = P + d.$$

Величина смещения подбирается так, чтобы число P_d было всегда положительным. В этом случае оказывается легче сконструировать математический сопроцессор для обработки вещественных чисел.

Для кодирования порядка в числах типа single используют смещение $d = 127_{10} = 7F_{16}$. Таким образом, для нашего примера

$$P_d = 100 + 111\,1111 = \mathbf{1000\,0011}.$$

Собирая теперь S, P_d и M в единое 32-разрядное число, получаем:

$$1\ \mathbf{10000011}\ 00010100000000000000000$$

или более компактно в шестнадцатеричной системе:

C1 8A 00 00.

Этот код и будет записан в память[1].

Не все двоичные комбинации для вещественных чисел соответствуют «правильным» числам: некоторые из них кодируют бесконечные значения, а некоторые — нечисловые данные (англ. **NaN** — *not a number*, «не число»). Они отличаются от остальных чисел тем, что имеют максимально возможный порядок[2] (например, для типа single это смещённый порядок $P_d = 255$, а для типа для double — 2047). Подобные «неправильные» данные возникают только в результате ошибок в вычислениях.

Таким образом, мы увидели, что целые и вещественные числа хранятся в памяти компьютера совершенно по-разному. Поэтому неудивительно, что свойства значений, скажем 3 и 3,0, в компьютерной арифметике совершенно разные.

---

[1] На самом деле в IBM-совместимых персональных компьютерах байты будут сохранены в памяти в обратном порядке: 00 00 8A C1.

[2] Это вполне логично, поскольку для вещественных чисел переполнение (получение «бесконечного» значения) наступает именно при больших порядках.

Вопросы и задания



1. Чем вызваны трудности, возникающие при представлении вещественных чисел в компьютере? Как они связаны с непрерывностью вещественных чисел в математике?
2. Объясните, как хранятся вещественные числа с фиксированной запятой. Почему этот метод не используется в современных компьютерах?
3. Что такое плавающая запятая? Из каких частей состоит число при кодировании с плавающей запятой?
4. Приведите примеры физических величин, которые обычно записывают в форме с плавающей запятой.
5. Почему метод представления чисел с плавающей запятой неоднозначен? Как изменится порядок, если запятую сместить на один разряд влево (вправо)?
6. Что такое нормализованная форма записи числа?
7. Как требования нормализации связаны с точностью представления вещественных чисел?
8. Единственно ли нормализованное представление числа? Все ли числа имеют нормализованное представление?
9. Почему старший бит значащей части нормализованного двоичного числа всегда равен единице? Как этот факт используется на практике?
10. Какие числа сохраняются в памяти с нулевой значащей частью?
11. На что влияет разрядность значащей части и разрядность порядка?
12. Почему задание разрядности для целых чисел однозначно определяет их свойства, а для вещественных — нет?
13. Что вы знаете о типах single, double и extended?
14. Как хранится порядок во всех рассмотренных форматах вещественных чисел? Почему не хранится знак порядка?
15. Сравните методы хранения отрицательных целых и вещественных чисел.
16. Как по двоичному представлению вещественного числа определить, положительное оно или отрицательное? Подходит ли этот метод для целых чисел?
17. В каком из вещественных форматов не используется «скрытая единица» и почему?
18. Какие логические операции и с какой маской надо применить, чтобы в переменной типа single:
    а) выделить значащую часть, сбросив порядок и знаковый бит;
    б) восстановить в полученной знаковой части «скрытую единицу»?
19. Как можно выделить смещённый порядок из числа типа single? Как получить истинное значение порядка?

20. С помощью какой маски можно выделить знаковый бит числа, хранящегося в формате single?
21. Что такое NaN?
22. Чем различаются представление в памяти целого числа и равного ему вещественного с нулевой дробной частью (например, 12 и 12,0)?



Подготовьте сообщение

а) «Типы данных для хранения вещественных чисел»
б) «Стандарт IEEE-754»



Задачи

1. Рассмотрим вымышленный 32-разрядный компьютер, в котором вещественные числа кодируются с фиксированной запятой, причём к целой части относится один байт, а к дробной — три. Рассчитайте для такой машины максимальное и минимальное допустимые числа и сравните с аналогичными значениями для типа *single*; объясните разницу. Вычислите также «порог» антипереполнения, т. е. минимальное число, отличное от нуля.
2. Запишите в нормализованном виде следующие десятичные вещественные числа: $43 \cdot 10^{21}$; 1040; 1,5; 0,32; 0,0005; $0{,}34 \cdot 10^{-12}$.
3. Запишите в нормализованном виде следующие двоичные вещественные числа (значащая часть и порядок даны в двоичной системе счисления): $11 \cdot 2^{10100}$; 10110; 1,1; 0,101; 0,0001; $11{,}001 \cdot 2^{-1000}$. Обратите внимание на значение первого бита значащей части.
4. Сравните диапазон чисел, который представляется в 32-битной форме *single*, с диапазоном целых 32-разрядных чисел со знаком.
5. Рассчитайте величину порядка со смещением для чисел типа *single* с двоичными порядками 11_2, 0 и 11_2.
6. Определите, как хранятся в памяти в формате single следующие вещественные десятичные числа: 1; 100; 0,1. Ответ запишите в шестнадцатеричной системе счисления.
7. Используя результаты предыдущей задачи, получите соответствующие коды для вещественных чисел –1, –100 и –0,1.
8. Определите, какому десятичному значению соответствуют коды (тип *single*): $3FC0\,00\,00_{16}$, $BFC0\,00\,00_{16}$, $3F44\,00\,00\,00_{16}$.

*9. Как преобразовать некоторое небольшое положительное вещественное число с нулевой дробной частью, например $9_{10} = 1{,}001_2 \cdot 2^{11}$, в форму беззнакового целого?

§ 30
Операции с вещественными числами

Сложение и вычитание

Рассмотрим принципы вещественной компьютерной арифметики на простых примерах. Сложим $7{,}25_{10} = 111{,}01$ и $1{,}75_{10} = 1{,}11$ (здесь и далее будем явно указывать систему счисления только для десятичных чисел). Представим эти числа в нормализованном виде: $111{,}01 \cdot 2^0 = 1{,}1101 \cdot 2^{10}$ и $1{,}11 \cdot 2^0$ (ещё раз подчеркнём, что значащие части и порядки чисел указываются в двоичной системе!). Не будем сейчас использовать «скрытую» единицу: это нужно только при сохранении чисел в памяти, а при изучении арифметических действий удобнее иметь «развёрнутые» значения.

Числа, записанные в форме с плавающей запятой, нельзя непосредственно сложить. Дело в том, что когда числа имеют разные порядки, их значащие части оказываются сдвинутыми друг относительно друга. Поэтому первое, что делает процессор перед сложением вещественных чисел, — **выравнивает** их **порядки** до большего. Число, имеющее меньший порядок p_2 (и значащую часть z_2), «подгоняется» к числу с большим порядком p_1 следующим образом:

1. Если $p_2 = p_1$, то порядки выровнены и преобразования закончены.
2. $p_2 = p_2 + 1$.
3. Сдвинуть значащую часть z_2 на один разряд вправо.
4. Перейти к шагу 1.

Для нашего примера разность порядков равна $10 - 0 = 10 = 2_{10}$, так что для выравнивания порядков значащую часть придётся сдвинуть дважды (порядок при этом увеличится на 2): $1{,}11 \cdot 2^0 = 0{,}0111 \cdot 2^{10}$. Подчеркнём, что ради проведения сложения нормализацию пришлось временно нарушить.

Теперь числа имеют одинаковый порядок и их **значащие части можно складывать:**

$$\begin{array}{r} 1{,}1101 \\ +\ 0{,}0111 \\ \hline 10{,}0100 \end{array}$$

Полный результат (с учётом порядка) равен $10{,}01 \cdot 2^{10}$ (убедитесь, что получившееся число равно 9_{10}). Но значащая часть результата больше 2, поэтому для записи числа в память его необходимо **нормализовать**: $10{,}01 \cdot 2^{10} = 1{,}001 \cdot 2^{11}$.

В этом примере мы нигде не учитывали ограниченность разрядной сетки и для простоты специально взяли короткие числа. Как же обстоит дело в реальных вычислениях? При выравнивании порядков происходит сдвиг значащей части меньшего из чисел вправо, при этом ее младшие (правые) разряды могут выйти за пределы разрядной сетки и будут отброшены. При сложении чисел с большой разностью порядков в результате таких сдвигов меньшее число может стать равно нулю. Например, представьте себе, что при 24-битной значащей части у слагаемых A и B разность порядков составляет, например, 26_{10}. В этом случае при выравнивании порядков произойдёт 26 сдвигов значащей части вправо, так что абсолютно все(!) её разряды исчезнут. В результате сложения окажется, что $A + B = A$, хотя $B \neq 0$ — это очередной (но далеко не единственный) пример погрешности компьютерных вычислений.

Умножение и деление

Числа, представленные в форме с плавающей запятой, «хорошо приспособлены» для выполнения умножения и деления. При **перемножении** достаточно (в полном соответствии с правилами математики) **перемножить** их **значащие части**, а **порядки сложить**. При **делении** значащие части **делятся**, а **порядки вычитаются**. Конечно, результат может оказаться ненормализованным, но это легко устраняется стандартной процедурой.

Рассмотрим, как выполняется умножение чисел $1{,}25_{10} = 1{,}01_2$ и $4{,}0_{10} = 100{,}0_2$. В нормализованном виде они запишутся как $1{,}01 \cdot 2^0$ и $100 \cdot 2^0 = 1{,}0 \cdot 2^{10}$. В этом примере значащие части можно перемножить устно: $1{,}01 \cdot 1{,}0 = 1{,}01$. Теперь сложим порядки: $0 + 10 = 10$. Таким образом, результат равен $1{,}01 \cdot 2^{10}$. Он уже удовлетворяет требованиям нормализации, поэтому никаких дополнительных действий не требуется. Легко показать, что $1{,}01 \cdot 2^{10} = 101 \cdot 2^0 = 5_{10}$.

Знакомство с вещественной арифметикой убедительно показывает важную роль битовых операций, изученных в § 28. Для нормализации постоянно используются сдвиги; для выделения значащей части или порядка из общего кода числа обязательно потребуется логическая операция «И», а для получения единого кода

из порядка и значащей части можно использовать логическое «ИЛИ». В обеих последних задачах также необходимы сдвиги.

Вопросы и задания

1. Почему перед сложением или вычитанием вещественных чисел требуется выравнивать порядки?
2. Какое число — большее или меньшее — подвергается сдвигу при выравнивании порядков? Почему?
3. Верно ли, что при выравнивании порядков значащая часть всегда сдвигается вправо?
4. Как вычислить количество сдвигов, которое потребуется произвести для выравнивания порядков?
5. Может ли получиться так, что при выполнении операции сложения значащие части придётся не складывать, а вычитать?
6. Как изменится двоичный код вещественного числа, если это число умножить на 2? Сравните с тем, как меняется код целого числа при удвоении.
7. Почему в компьютерной арифметике возможны случаи, когда $A + B = A$ при $B \neq 0$? При каких условиях может так получиться?

*8. Может ли при вычитании быть переполнение? Антипереполнение?

9. Сформулируйте правила умножения и деления вещественных чисел.

*10. Верно ли, что когда для размещения результата умножения в значащей части не хватает разрядов — это переполнение? Когда возникает переполнение?

11. Может ли в результате арифметической операции нарушиться нормализация? Как в таких случаях нужно поступать? Приведите пример.

Задачи

1. Суммируются два двоичных числа: $1{,}0 \cdot 2^{100}$ и $1{,}11 \cdot 2^{11}$. Какое из них будет сдвигаться при выравнивании порядков? На сколько разрядов?
2. Выполните сложение двух десятичных чисел 2,5 и 0,125, предварительно преобразовав их к двоичной форме и выделив значащую часть и порядок. Результат представьте в формат single и запишите в шестнадцатеричном коде.
3. Выполните вычитание десятичных чисел 0,125 – 2,5, предварительно преобразовав их к двоичной форме и выделив значащую часть и порядок. Вычтите из большего числа (2,5) меньшее (0,125), а знак добавьте в конце. Результат представьте в формате *single* и запишите в шестнадцатеричном коде.

*4. Выполните сложение двух десятичных чисел 0,1 и 0,2, предварительно преобразовав их к двоичной форме и выделив значащую часть и порядок. Предположите, что на значащую часть выделяется 8 разрядов, «скрытую единицу» не используйте.

5. Докажите, что если двоичное число $1{,}01 \cdot 2^{10}$ сложить с самим собой, то его значащая часть не изменится, а порядок возрастёт на 1.

*6. Заданы 5 вещественных чисел, причём одно из них $A = 1{,}0 \cdot 2^{111}$, а все остальные равны между собой: $B = C = D = E = 1{,}0 \cdot 2^{-10}$ (значения значащих частей и порядков записаны в двоичной системе счисления). Убедитесь, что в случае, когда значащая часть представлена 8 битами («скрытая единица» не используется), результат сложения всех чисел оказывается зависящим от порядка сложения, в частности $A + B + C + D + E \neq E + D + C + B + A$. Объясните результат.

7. Перемножьте два десятичных числа 0,75 и 1,25, предварительно преобразовав их к двоичной форме и выделив значащую часть и порядок. Для проверки переведите результат в десятичную систему счисления.

8. Двоичное число $1{,}1 \cdot 2^{1000}$ записано в формате, в котором под порядок вещественного числа отводятся 4 бита. Произойдёт ли переполнение, если число возвести в квадрат?

9. Выполните деление десятичных чисел 0,75 и 0,25, предварительно преобразовав их к двоичной форме и выделив значащую часть и порядок. Для проверки переведите результат в десятичную систему счисления.

10. Двоичное число $1{,}0 \cdot 2^{-1000}$ записано в формате, в котором под порядок вещественного числа отводятся 4 бита. Что произойдёт, если его разделить на $1{,}0 \cdot 2^{1001}$?

Практические работы к главе 4

Работа № 9 «Целые числа в памяти»
Работа № 10 «Арифметические операции»
Работа № 11 «Логические операции и сдвиги»

ЭОР к главе 4 на сайте ФЦИОР (http//fcior.edu.ru)

- Дополнительный код числа. Алгоритм получения дополнительного кода отрицательного числа
- Число и его компьютерный код
- Числа с фиксированной и плавающей запятой

Самое важное в главе 4

- Для хранения чисел в памяти компьютера используется конечное число разрядов. Из-за этого числа в компьютере имеют ограниченный диапазон, а результаты вычислений могут быть неточными.
- Для хранения целого числа может быть использовано 8, 16, 32 или 64 бита памяти. Каждый дополнительный бит расширяет диапазон допустимых чисел в 2 раза.
- Отрицательные целые числа хранятся в дополнительном двоичном коде, который позволяет выполнять вычисления с положительными и отрицательными числами по одному и тому же алгоритму.
- Вещественные числа хранятся в памяти компьютера в формате с плавающей запятой: отдельно значащая часть и порядок. Из-за ограниченности числа разрядов вещественное число, как правило, не удаётся точно представить в памяти.
- При вычислениях с вещественными числами накапливаются ошибки. Для повышения точности расчётов нужно стараться, если возможно, использовать только операции с целыми числами.

Глава 5
Устройство компьютера

Компьютер — это универсальный программируемый автомат для обработки данных.

Из этого определения можно сделать вывод, что компьютер состоит из двух важнейших составляющих: **аппаратной части** и **программного обеспечения** (ПО). В технической литературе их часто называют английскими терминами **hardware** и **software**[1].

Поскольку одно и то же оборудование может быть перенастроено на выполнение новых задач простой заменой ПО, такие универсальные компьютеры можно выпускать большими партиями, и это делает их производство проще и дешевле. За счёт этого во многих областях они заменили специализированные устройства.

Исторически существовали два принципиально разных типа вычислительных машин — *аналоговые* и *цифровые*. Они различались по способу представления обрабатываемых данных: в аналоговой или цифровой форме. Цифровая техника быстро доказала свои преимущества:

- высокую точность вычислений;
- универсальность и быстроту перехода от одной задачи к другой;
- способность хранить большие объёмы данных.

В результате почти все современные компьютеры работают только с дискретной (цифровой) информацией. Поэтому в этой главе рассматривается только цифровая вычислительная техника.

---

[1] Слово «hardware» означает металлические изделия, а применительно к компьютеру — его детали (платы, монитор и прочее «железо»). Термин «software» исключительно компьютерный, он возник из противопоставления слов «soft» (мягкий, гибкий, податливый) и «hard» (твёрдый, жёсткий, негнущийся). Это значит, что software гибко «подстраивает» hardware для решения разнообразных задач.

§ 31 История развития вычислительной техники

История появления и развития вычислительной техники — это обширная тема, которой посвящено множество книг. С точки зрения курса информатики, исторические сведения интересны, прежде всего, тем, что позволяют отследить основные направления развития компьютерной техники и попытаться предвидеть её ближайшее будущее.

В отечественной технической литературе приблизительно до 80-х годов прошлого века везде использовался термин **«электронная вычислительная машина» (ЭВМ)**. Позднее это словосочетание стало постепенно вытесняться новым более коротким названием **«компьютер»**. Все разновидности современной вычислительной техники сейчас называются только компьютерами, но, тем не менее, старые модели по традиции именуются ЭВМ.



Блез Паскаль (1623–1662)

Вехи истории

Первая **механическая машина**, с помощью которой можно было производить вычисления, была изготовлена известным французским учёным *Блезом Паскалем* в 1645 г. Чтобы отдать дань уважения этому изобретению, один из языков программирования впоследствии был назван именем *Паскаль*.

Идея о реализации вычислений в автоматическом режиме (без участия человека) впервые была предложена и детально развита английским учёным *Чарльзом Бэббиджем*. Он спроектировал и описал **Аналитическую машину**, состав и принципы действия которой фактически повторились в будущих ЭВМ.



Чарльз Бэббидж (1791–1871)



Ада Лавлейс (1815–1852)

Бэббидж посвятил всю свою жизнь работе над машиной, но построить её из механических деталей не удалось: уровень техники XIX века не позволял изготовить столь сложный и точный механизм.

Разработкой принципов программирования Аналитической машины Бэббиджа занималась *Ада Лавлейс* (дочь английского поэта Д. Г. Байрона). Её идеи оказали большое влияние на развитие программирования. Например, ей принадлежат термины «цикл» и «рабочая ячейка». В честь первого в мире программиста один из языков программирования получил имя *Ада*.

Первой ЭВМ, продемонстрировавшей на практике возможность автоматических расчётов по программе, считается **ЭНИАК** (сокращение от английского словосочетания Electronic Numeric Integrator and Computer) — рис. 5.1. Эта ЭВМ была построена в 1944 г. в США под руководством *Джона Моучли*; главным инженером проекта был *Преспер Эккерт*. ЭНИАК содержал 18 000 электронных ламп и, занимая зал 9 × 15 м^2, потреблял около 150 кВт электроэнергии; выполнял более 350 умножений и 5000 сложений за секунду. Данные вводились в машину с помощью перфокарт, а программа обработки набиралась с помощью штекеров на специальных панелях.

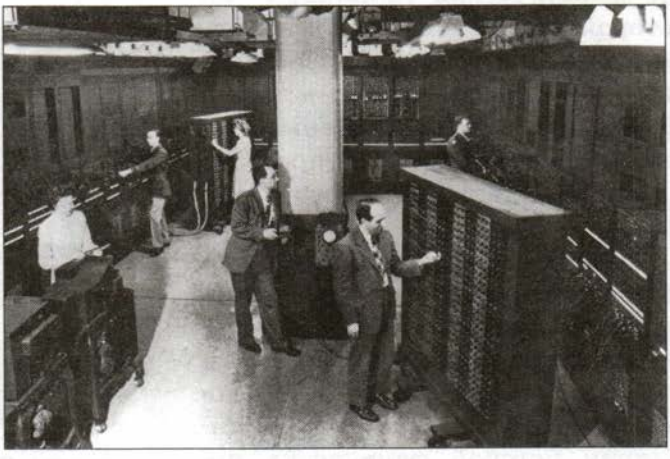


Рис. 5.1. ЭВМ «ЭНИАК» (www.fi.edu)

Опыт построения первой ЭВМ был проанализирован *А. Берксом*, *Г. Голдстайном* и *Дж. фон Нейманом*. В 1946 г. они опубликовали работу «Предварительное рассмотрение логической конструкции электронного вычислительного устройства», ставшую классической. Сформулированные в ней принципы построения вычислительных машин используются и сейчас, несмотря на то что со времени публикации прошло более полувека. В компьютерной литературе эти принципы часто называют **фон-неймановской архитектурой ЭВМ**, хотя Джон фон Нейман не был её единоличным автором.



Дж. фон Нейман (1903–1957)

Дальнейший прогресс вычислительной техники во многом определялся развитием её **элементной базы**. Важной вехой на этом пути стало создание в 1947 г. **транзистора** (*У. Шокли*, *Д. Бардин* и *У. Браттейн*). Транзистор — это полупроводниковый прибор для управления электрическими сигналами (рис. 5.2). На основе транзисторов могут собираться цифровые электронные схемы, такие как логические элементы и триггеры.

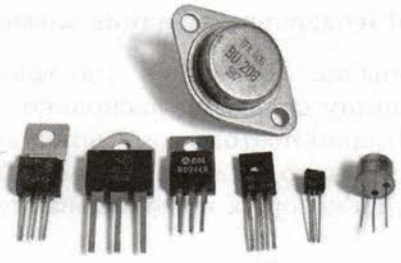


Рис. 5.2. Транзисторы

В 1958 г. *Дж. Килби* разработал первую **интегральную микросхему** — кристалл, в котором размещается не один транзистор, а целая схема на нескольких транзисторах — например, один или несколько триггеров. Все её вспомогательные радиодетали (резисторы, конденсаторы и другие) также изготавливаются с помощью полупроводниковых технологий.

Первый **микропроцессор** Intel 4004 был разработан под руководством инженера *М. Хоффа* и выпущен в 1971 г. Он был че-



С. Джобс и С. Возняк с компьютером Apple-I (cedmagic.com)

тырёхбитным, поскольку предназначался для микрокалькуляторов и должен был производить вычисления над десятичными числами (см. § 14, где рассказано о двоично-десятичном кодировании). Интересно, что цель его создания была чисто технологическая: заменить 12 специализированных микросхем одной универсальной, но программируемой. Достоинства микропроцессора как главного узла для компьютеров были оценены позднее.

В ходе совершенствования элементной базы вычислительные машины становились все более мощными и компактными. В результате появились персональные компьютеры (ПК), которыми мы сейчас пользуемся. В 1976 г. два молодых приятеля *С. Джобс* и *С. Возняк* в гараже родителей Джобса собрали ПК **Apple**, положивший начало известному ныне семейству компьютеров. А в 1981 г. был продемонстрирован первый компьютер другого семейства — **IBM PC** (IBM Personal Computer), потомки которого в нашей стране особенно широко распространены.

Поколения ЭВМ (совершенствование элементной базы)

Неудачная попытка Бэббиджа построить механическую Аналитическую машину показала, насколько важную роль играет элементная база. Именно поэтому дальнейшую историю вычислительной техники принято делить на периоды в соответствии с теми элементами, из которых изготавливались машины.

Первое поколение ЭВМ относят к периоду примерно 1945–1955 гг. Эти машины были построены на базе **электронных ламп**[1]. Открыл его уже описанный ранее **ЭНИАК**. В нашей стране машинами первого поколения были **МЭСМ** (Малая электронная счётная машина, 1951 г., рис. 5.3), **БЭСМ** (Большая, или Быстродействующая, электронная счётная машина, 1952 г.), **Стрела**

[1] В середине XX века было разработано несколько счётных машин на электромагнитных реле, но их из-за малого количества не принято включать в классификацию поколений.

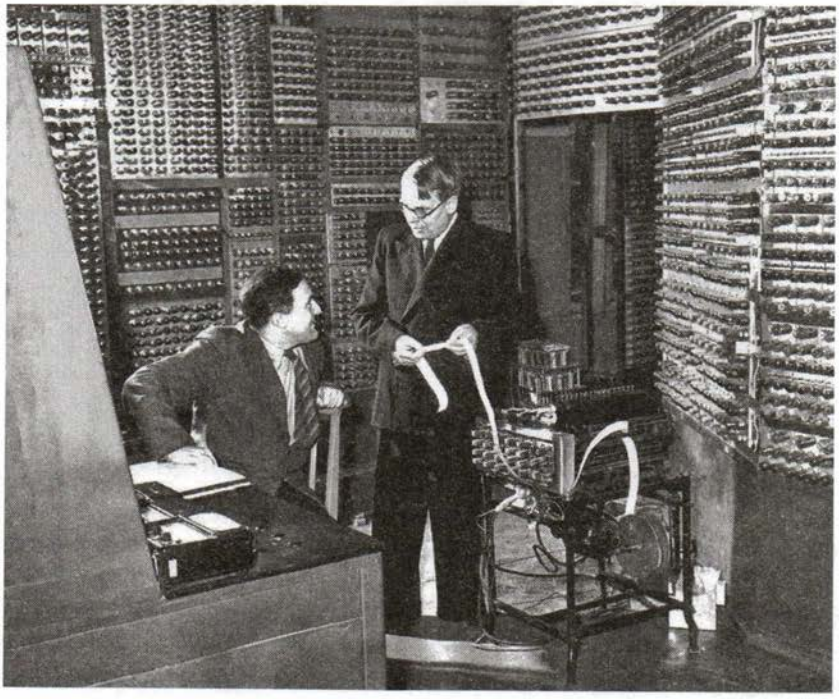


Рис. 5.3. ЭВМ первого поколения МЭСМ
(фото из Единой коллекции цифровых образовательных ресурсов)

(1953 г.), **Урал** (1954 г.), **М-20** (1959 г.). Все эти машины были огромными, неудобными и дорогими.

Второе поколение ЭВМ (примерно 1955–1965 гг.) появилось, когда на смену лампам в электронных схемах пришли **транзисторы.** Первый экспериментальный компьютер на транзисторах **ТХ-0** был создан в 1955 г. в Массачусетском технологическом институте (США). ЭВМ на транзисторах были значительно меньше и имели существенно более высокое быстродействие; они потребляли гораздо меньше энергии, были надёжнее и не требовали таких громоздких систем отвода тепла, как ламповые машины. Многие машины второго поколения уже помещались в обычной комнате среднего размера, например ЭВМ серии **Наири** (1964 г.) или **МИР** (Машина инженерных расчётов, 1965 г.). Наиболее производительными ЭВМ этого поколения стали **Стретч** (США, 1960 г.), **Атлас** (Великобритания, 1961 г.), **CDC 6600** (США, 1964 г.) и **БЭСМ-6** (СССР, 1967 г., рис. 5.4).



Рис. 5.4. ЭВМ второго поколения БЭСМ-6
(фото из Единой коллекции цифровых образовательных ресурсов)

Третье поколение ЭВМ (примерно 1965–1975 гг.) связано с появлением **интегральных микросхем**. Размеры элементов, из которых строились вычислительные машины, существенно уменьшились (рис. 5.5). Казалось бы, размеры самих этих ЭВМ снова должны были существенно уменьшиться, но этого не произошло. Дело в том, что ЭВМ третьего поколения были предназначены для коллективной (многопользовательской) работы. Это было время крупных вычислительных центров, предоставлявших услуги огромному числу пользователей из многих организаций. Поэтому главное внимание уделялось не уменьшению размеров и стоимости машин, а повышению их вычислительной мощности и эффективности обработки больших объёмов данных.

Отличительная черта третьего поколения — выпуск семейств вычислительных машин, которые были совместимы между собой как аппаратно (все устройства сконструированы по одинаковым стандартам), так и программно (имели одинаковую систему команд). Впервые идею общей **архитектуры**, обеспечивающей выполнение написанных ранее программ на любой новой модели, предложила фирма *IBM*, которая разработала семейства больших ЭВМ **IBM/360** и **IBM/370**. В этот период в СССР было принято решение перейти к копированию зарубежной техники ради обеспечения совместимости. В результате в странах Восточной Европы

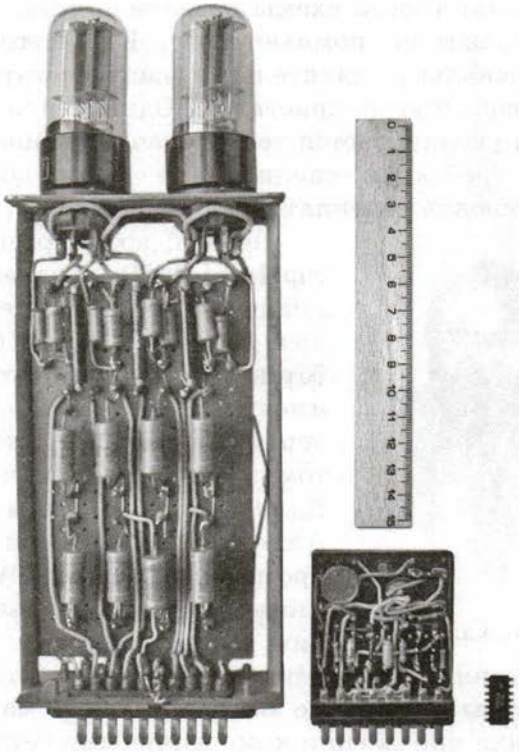


Рис. 5.5. Уменьшение размеров ячеек памяти ЭВМ от первого до третьего поколений (в каждой из них — по два триггера)

были выпущены «аналоги» упомянутых выше семейств под общим названием **ЕС ЭВМ** (Единая система ЭВМ). Одновременно появились мини-ЭВМ семейства **СМ ЭВМ** (Семейство малых ЭВМ), аналогичное известному зарубежному семейству **PDP** фирмы **DEC**.

Четвёртое поколение ЭВМ берёт своё начало примерно с 1975 г. Прогресс в электронике дал возможность существенно увеличить плотности «упаковки» элементов на кристалле, и в одной микросхеме теперь удавалось собрать целый узел, например микропроцессор. Микросхемы такого уровня стали называть **БИС** (большие интегральные схемы, от 1000 до 10 000 элементов на кристалле), а позднее — **СБИС** (сверхбольшие ИС, более 10 000 элементов). Именно они стали основой четвёртого поколения ЭВМ, которое существует вплоть до настоящего времени.

Увеличение плотности схемы позволило в первую очередь повысить быстродействие компьютеров[1]. Кроме того, возросла и надёжность, поскольку значительная часть электрических соединений выполнена внутри кристалла. Однако при высокой плотности монтажа увеличивается теплоотдача от миниатюрных деталей, поэтому требуются специальные меры по отводу тепла (например, установка вентиляторов охлаждения).



Б. Гейтс и П. Аллен

Первый восьмиразрядный процессор **Intel 8080**, предназначенный специально для компьютеров, был выпущен в 1974 г. На его базе был разработан микрокомпьютер **Альтаир**, имевший большой коммерческий успех. Он вошел в историю ещё и потому, что в 1975 г. молодой студент *Билл Гейтс* со своим другом *Полом Алленом* реализовали на нём язык программирования **BASIC**. Чуть позднее они создали известную компанию *Microsoft*.

Кроме **персональных компьютеров** к четвёртому поколению относятся **серверы** — мощные вычислительные машины, которые используются для управления компьютерными сетями. Они предоставляют свои ресурсы (например, принтеры, файлы или программы) в коллективное пользование. Серверы могут эффективно обслуживать большое количество пользователей одновременно. Например, два сервера *Hewlett-Packard T600* (по 12 процессоров в каждом), установленные в системе резервирования билетов *Amadeus*, способны практически без задержек обслуживать примерно 60 миллионов запросов в сутки (система имеет около 180 тысяч терминалов в более чем ста странах мира).

Важное направление в компьютерах четвёртого поколения — **параллельная** (одновременная) **обработка данных**. Если решаемую задачу удается разбить на независимые друг от друга действия, то их необязательно делать друг за другом, а можно для экономии времени выполнять одновременно. Правда, для этого

---

[1] При быстродействии 10^9 элементарных операций в секунду (типичное по порядку величины значение для современного компьютера) за время каждой из них электрический сигнал со скоростью $3 \cdot 10^8$ м/с успевает пройти путь всего 30 см.

требуется несколько процессоров, но современный уровень техники это позволяет. Более того, в последнее время были сконструированы **многоядерные процессоры**, т. е. фактически несколько процессоров в одном кристалле.

Мощные многопроцессорные компьютеры, в которых выполняется параллельная обработка данных, называют **суперкомпьютерами**. Это уникальные устройства, поэтому они изготавливаются штучно.

В литературе часто упоминаются суперкомпьютеры серии **CRAY**, разработанные под руководством *Сеймура Крэя*. Первая модель этой серии, *CRAY-1*, была построена в США в 1976 г. и имела огромный коммерческий успех.

Все развитые страны ведут жёсткую конкуренцию в области суперкомпьютеров, поскольку обладание такой техникой позволяет решать стратегически важные вычислительные задачи:

- исследование геофизики Земли, прогнозирование изменений климата на планете;
- создание математических моделей молекул (полимеров, кристаллов и т. п.), синтез новых материалов и лекарств;
- расчёт процессов горения и взрыва, а также моделирование других физических задач (обтекание летательных аппаратов, прочность кузовов автомобилей);
- моделирование и прогнозирование ситуации в экономике;
- расчёты процессов нефте- и газодобычи, а также сейсморазведки недр;
- проектирование новых электронных устройств.

Приведём несколько примеров применения суперкомпьютеров. Исследователи фирмы IBM на протяжении десятилетий изучают деятельность мозга и пытаются моделировать её. В 2009 г. появилось сообщение о том, что полученная в рамках проекта модель мозга по своим возможностям превзошла уровень кошки: моделируется 1 миллиард нейронов и 10 триллионов связей между ними! Модель работает на базе суперкомпьютера *Blue Gene/P*, имеющего 147 456 процессоров и 144 Тбайт оперативной памяти.

По данным компании *Ford Motor*, благодаря детальному моделированию на суперкомпьютере, количество разрушаемых в крэш-тестах[1] автомобилей удаётся сократить на треть.

---

[1] Крэш-тесты — это тесты, исследующие поведение машин при сильном ударе о бетонное препятствие.

Применение суперкомпьютеров для расчёта состава лекарств позволяет уменьшить срок их разработки с нескольких лет до полугода.

Мощные компьютеры используются при создании компьютерных спецэффектов в кино. Например, для фильма «Властелин колец» фирме *WETA Digital* потребовалось столько суперкомпьютеров, что Новая Зеландия вышла на первое место в мире по их количеству на душу населения.

В 2009 г. в МГУ введён в строй самый мощный **российский суперкомпьютер «Ломоносов»** (рис. 5.6) производительностью около 400 Тфлопс[1]. В его состав входят 8892 многоядерных процессора (общее число ядер — 35 776). На момент запуска «Ломоносов» занимал в мировом рейтинге суперкомпьютеров Top500 двенадцатое место.

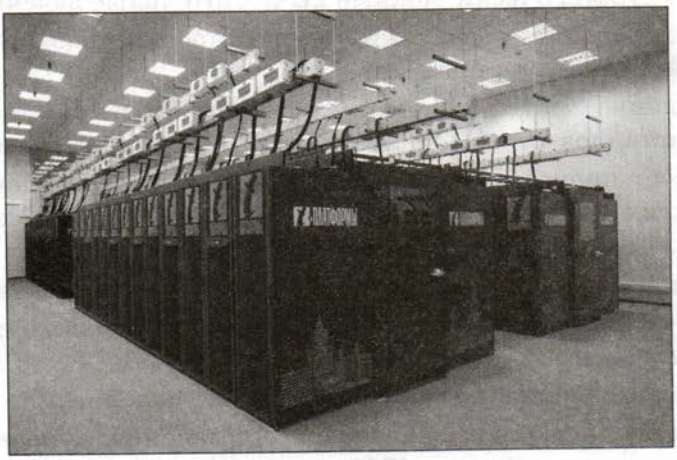


Рис. 5.6. Суперкомпьютер «Ломоносов»[2]

Много шума наделал *японский проект по созданию компьютеров пятого поколения* (1982–1992 гг.). Было заявлено, что в основе компьютеров пятого поколения будут уже не вычисления,

---

1 Флопс (англ. *FLOPS — floating point operations per second*) — единица измерения количества операций с вещественными числами за 1 секунду; приставка «тера» добавляется по тем же правилам, что и при измерении информации; очевидно, что операции над вещественными числами намного сложнее и выполняются гораздо дольше, чем над целыми.

2 Фотография предоставлена компанией «Т-Платформы» (www.t-platforms.ru).

а логические заключения, т. е. произойдёт переход от обработки данных к обработке знаний. Машину обещали научить воспринимать речевые команды человека, читать рукописный текст, анализировать графические изображения и делать многие другие нетривиальные для компьютера вещи. Планы проекта были грандиозны. Но, несмотря на щедрое финансирование и передовые позиции японских технологий, успехи оказались весьма скромными. На основе программного моделирования на компьютерах четвёртого поколения удалось реализовать лишь отдельные детали проекта, причём реальная машина, работающая на базе логических выводов, так и не вышла за стены лабораторий.

Таким образом, создание принципиально новых компьютеров пятого поколения закончилось неудачей. Все компьютеры, используемые в настоящее время, по-прежнему построены на базе идей четвёртого поколения. Классификация поколений «замерла» в ожидании новых революционных идей. Такие идеи особенно необходимы ещё и потому, что электронная техника уже подошла к пределу быстродействия, который определяется законами физики: для увеличения скорости передачи данных требуется уменьшать размеры электронных деталей, но плотность упаковки транзисторов в полупроводниковом кристалле и так уже практически достигла максимально возможной. Поэтому *идёт поиск неэлектронных средств хранения и обработки данных*.

В первую очередь учёные попытались использовать в качестве носителя информации свет — так появились **оптические процессоры**. В них можно применять параллельную обработку данных, например одновременно выполнять какую-то операцию со всеми пикселями изображения. В 2003 г. был выпущен оптический процессор *Enlight256*, который имеет оптическое ядро, а входные и выходные данные представлены в электронном виде. Быстродействие этого процессора — 8 триллионов операций в секунду. Он состоит из 256 лазеров, набора линз и фотоприёмников. Оптические процессоры используются в военной технике и при обработке видеоданных в реальном времени.

Большие надежды связаны с **разработкой квантовых компьютеров**, в которых применяются идеи квантовой физики, описывающей законы микромира и поведение отдельных элементарных частиц. Данные для обработки в квантовом компьютере записываются в систему *кубитов* — квантовых битов. Затем с помощью специальных операций состояние этой системы изменяют по определённому алгоритму. Конечное состояние системы куби-

тов — это и есть ответ в задаче. Особые свойства кубитов[1] позволяют организовать параллельную обработку данных, так же как и в многопроцессорных системах. Поэтому многие задачи, для решения которых сейчас не хватает вычислительных ресурсов (например, раскрытие шифров), будут достаточно быстро решены, как только квантовый компьютер будет построен.

В некоторых лабораториях ведётся **разработка биологических компьютеров** (биокомпьютеров), которые работают как живой организм. Ячейки памяти биокомпьютеров — это молекулы сложных органических соединений, например молекулы ДНК, в которых хранится наследственная информация. Сам процесс вычислений — это химическая реакция, результат — состав и строение получившей молекулы.

Проводятся также исследования и в области **нанотехнологий**, с помощью которых планируют построить транзистор размером с молекулу.

Развитие возможностей от поколения к поколению

С каждым поколением вычислительных машин развивались их **аппаратные возможности**. ЭВМ становились более мощными и универсальными. Расширялось количество обрабатываемых **типов данных**:

первое поколение — только числовые данные;
второе поколение — числа и символы;
третье поколение — числа, текстовые и графические данные;
четвёртое поколение — добавились аудио- и видеоданные.

Появившись как устройство для облегчения вычислений, компьютер сейчас всё более активно обрабатывает разнообразную нечисловую информацию. Чтобы подчеркнуть широкие возможности современных компьютеров, введён специальный термин «мультимедиа».

Мультимедиа (от латинских слов *multum* — много и *medium* — средства) — одновременное использование различных форм представления информации (графика, текст, видео, фотографии, анимация, звук и т. д.) и их объединение в одном объекте.

[1] В отличие от привычного нам бита кубит устроен так, что способен вместить в себя гораздо больше информации.

Термин «мультимедиа» также используется в словосочетаниях «мультимедиа компьютер», «мультимедиа носитель», «программные и аппаратные средства мультимедиа». Пример мультимедиа объекта — компьютерная презентация.

Как правило, при использовании технологий мультимедиа человек может влиять на показ материалов: перейти вперёд или вернуться назад, изменить настройки, выбрать один из предложенных вариантов и т. п. Подобное взаимодействие человека и компьютера называют **интерактивностью** (взаимной активностью).

Другое направление в развитии аппаратной части — это увеличение разнообразия и одновременно рост сложности **внешних устройств**, присоединяемых к ЭВМ:

первое поколение — штекеры и переключатели, индикаторные лампочки, устройства ввода с перфокарт;

второе поколение — перфоленты, магнитные ленты и барабаны, печатающие устройства;

третье поколение — магнитные диски, текстовые и графические мониторы, графопостроители;

четвёртое поколение — огромное разнообразие внешних устройств, в том числе:

- устройства для хранения данных на оптических дисках;
- мышь, джойстик, шлемы виртуальной реальности и др.;
- возможность подключения бытовой электроники (фотоаппаратов, музыкальных плееров, сотовых телефонов и др.) с помощью кабелей и беспроводных соединений.

Каждое новое поколение компьютеров расширяет возможности **программного обеспечения (ПО)**. Использовать современный компьютер без ПО практически невозможно. Стоимость ПО нередко намного превышает стоимость аппаратной части (в первых поколениях было наоборот!).

Первое поколение. Программы разрабатывали хорошо подготовленные специалисты на машинном языке, сама программа представляла собой последовательность чисел (машинных кодов). Стандартного программного обеспечения практически не было.

Второе поколение. Появились первые языки программирования. Некоторые из них были разработаны для конкретных машин, но значительно удобнее оказались машинно-независимые языки, такие как **Фортран** (1957) и **Алгол** (1960). Написать программу на таком языке было значительно проще: с этим уже

вполне мог справиться рядовой научный работник, причем не обязательно с математическим образованием. В конце второго поколения появились **специальные программы**, управляющие последовательным прохождением заданий через ЭВМ в автоматическом режиме (они назывались **мониторами**). Их дальнейшее развитие привело к появлению операционных систем.

Третье поколение. Созданы **операционные системы** (ОС), которые обеспечивали работу компьютеров в многопользовательском режиме и управляли большим количеством сложных внешних устройств (в первую очередь магнитными дисками). Для «общения» с ОС разработаны специальные языки управления заданиями. Широкое распространение получили созданные ранее языки программирования, например **Фортран** для математических вычислений и **Кобол** для экономических расчётов. Начали появляться пакеты прикладных программ для решения задач в конкретных областях.

Четвёртое поколение. Для управления компьютером пользователь теперь использует не язык программирования, а различные меню и кнопки. Например, необходимую команду можно выбрать из меню (перечня доступных в данной ситуации возможностей) на естественном языке. Стало реально освоить компьютер после очень короткой подготовки. Программное обеспечение для ПК становится необычайно разнообразным — написано столько программ, что их трудно даже просто систематизировать. Казалось бы, такое разнообразие ПО должно сделать программирование ненужным, но нередко проще написать собственную небольшую программу решения конкретной задачи, чем тратить время на поиск и освоение возможностей готовых универсальных пакетов.

Таким образом, при переходе от поколения к поколению возрастает вычислительная мощность компьютеров. Значительная часть новых возможностей направляется на повышение удобства работы пользователя. В человеко-машинном общении отчётливо прослеживается движение от машинного языка к языкам, естественным для человека. В результате расширяется область применения и круг пользователей компьютерной техники.

Вопросы и задания

1. Что такое компьютер?
2. Охарактеризуйте программную и аппаратную части компьютера.

3. Почему универсальный компьютер с изменяемой программой удобнее, чем специализированная техника? Ответ обоснуйте.
4. Что такое цифровая и аналоговая техника?
5. Почему цифровая техника вытеснила аналоговую?
6. Перечислите основные вехи в истории развития вычислительной техники.
7. Какова заслуга Чарльза Бэббиджа?
8. В честь кого названы языки программирования Ада и Паскаль? Какое отношение эти люди имеют к вычислительной технике?
9. Что такое транзистор и микросхема? Из чего они изготавливаются?
10. С какой целью разрабатывались первые микропроцессоры?
*11. Почему микропроцессор Intel 4004 был специально спроектирован для работы только с четырёхбитными данными? Указание: вспомните, как можно хранить отдельные десятичные цифры числа.
12. По какому принципу ЭВМ делятся на поколения?
13. Почему время существования того или иного поколения всегда указывается приблизительно?
14. Перечислите все поколения ЭВМ и назовите элементную базу каждого из них.
15. Что даёт уменьшение базовых элементов вычислительной техники?
16. Почему электронные схемы требуют охлаждения? Все ли элементы нуждаются в дополнительном охлаждении?
17. Какие поколения вычислительных машин построены на базе полупроводниковых технологий? Чем отличается друг от друга их элементная база?
18. Объясните, почему большинство ЭВМ третьего поколения имели крупные габариты, несмотря на очередное уменьшение размеров элементной базы.
19. Когда появились первые семейства ЭВМ? Какая фирма предложила идею? В чем преимущества выпуска совместимых моделей?
20. Компьютеры какого поколения сейчас стоят на полках магазинов?
21. Какие разновидности компьютеров входят в четвёртое поколение?
22. Как вы понимаете термин «персональный компьютер»?
23. Какие семейства персональных компьютеров вы знаете? Какое из них появилось раньше?
24. Перечислите бытовые приборы, в которых применяются микропроцессоры.
25. Что такое суперкомпьютеры? Зачем они используются?
26. Найдите в Интернете рейтинг суперкомпьютеров Top500. Какие страны занимают в нём лидирующее положение? Есть ли там российские компьютеры?
*27. Зачем в суперкомпьютерах так много процессоров? Подумайте, любая ли задача может быть решена быстрее, если её считать параллельно на множестве процессоров? (В качестве помощи можно вос-

пользоваться аналогией с распределением частей одного большого задания между учениками класса.)

28. Назовите примеры вычислительных машин каждого из четырёх поколений. Найдите в Интернете дополнительный материал об этих машинах.
29. Что вы можете сказать о судьбе пятого поколения компьютеров?
*30. Почему, по-вашему мнению, уже довольно давно не происходило смены поколений?
31. Данные каких типов обрабатывались на ЭВМ каждого из поколений?
32. Как изменялся набор внешних устройств при переходе от одного поколения к другому?
33. Опишите, как происходило развитие программного обеспечения.
34. Что вы можете сказать по поводу роли программного обеспечения: уменьшается она или увеличивается по сравнению с предыдущими поколениями?
35. Предположим, что появился процессор с каким-то принципиально новым свойством. Как быстро этим свойством смогут воспользоваться потребители? Какова роль программного обеспечения в этом?
36. Быстродействие вычислительной техники постоянно растёт. Как же тогда объяснить, что пользователи жалуются на «медлительные» компьютеры и все время стараются купить новые, ещё более производительные?
*37. Влияет ли развитие программных средств на развитие аппаратной части?
38. Что представляли собой программы для первых машин? Почему для их записи было удобно использовать не двоичную систему счисления, а восьмеричную или шестнадцатеричную?
39. Зачем были созданы языки программирования? Когда они появились?
40. Попробуйте назвать положительные и отрицательные последствия огромного разнообразия существующих программ.
41. Почему развитие ПО расширяет количество пользователей компьютера?
42. Когда появились операционные системы и с чем это связано?
*43. Насколько сейчас, по-вашему, актуально умение программировать? Попробуйте найти аргументы «за» и «против» (учитывайте разные цели работы на компьютере у людей).

Подготовьте сообщение

а) «Что такое микропроцессор?»
б) «Физические пределы быстродействия компьютеров»
в) «Много программ — это хорошо или плохо?»
г) «Зачем нужно программировать?»

Подготовьте доклад

а) «Первые ЭВМ»
б) «Поколения ЭВМ»
в) «Программное обеспечение и поколения ЭВМ»
г) «Разработка компьютеров будущего»
д) «Квантовые компьютеры»
е) «Суперкомпьютеры»

§ 32
Принципы устройства компьютеров

В предыдущем параграфе вы увидели, что вычислительная техника в своем развитии прошла целый ряд характерных этапов. Несмотря на это, некоторые фундаментальные (базовые, основные) принципы устройства ЭВМ почти не изменились. Поэтому логично начать знакомство с устройством компьютера именно с них.

Классические принципы построения ЭВМ были предложены в работе *А. Беркса, Г. Голдстайна и Дж. фон Неймана «Предварительное рассмотрение логической конструкции электронного вычислительного устройства»*. Обычно выделяют[1] следующие наиболее важные идеи этой работы:

- состав основных компонентов вычислительной машины;
- принцип двоичного кодирования;
- принцип адресности памяти;
- принцип иерархической организации памяти;
- принцип хранимой программы;
- принцип программного управления.

Рассмотрим их подробнее.

Общие принципы

Основные компоненты машины. В самом первом разделе с таким названием фон Нейман с соавторами определили и обосновали состав ЭВМ:

«Так как законченное устройство будет универсальной вычислительной машиной, оно должно содержать несколько основных органов, таких как орган арифметики, памяти, управле-

[1] Эта техническая статья не содержит отдельного пронумерованного перечня принципов, поэтому в учебной литературе встречаются непринципиальные отличия в их формулировке и описании.

ния и связи с оператором. Мы хотим, чтобы машина была полностью автоматической, т. е. после начала вычислений работа машины не зависела от оператора».

Таким образом, ЭВМ должна состоять из нескольких блоков, каждый из которых выполняет вполне определённую функцию. Эти блоки есть и в сегодняшних компьютерах:

- **арифметико-логическое устройство** (АЛУ), в котором выполняется обработка данных;
- **устройство управления** (УУ), обеспечивающее выполнение программы и организующее согласованное взаимодействие всех узлов машины; сейчас АЛУ и УУ изготавливают в виде единой интегральной схемы — микропроцессора;
- **память** — устройство для хранения программ и данных; память обычно делится на **внутреннюю** (для временного хранения данных во время обработки) и **внешнюю** (для длительного хранения между сеансами обработки);
- **устройства ввода**, преобразующие входные данные в форму, доступную компьютеру;
- **устройства вывода**, преобразующие результаты работы ЭВМ в форму, удобную для восприятия человеком.

В классическом варианте все эти устройства взаимодействовали через процессор (рис. 5.7).

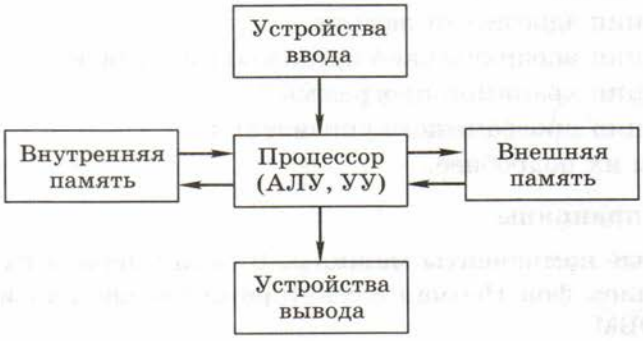


Рис. 5.7

Принцип двоичного кодирования. Устройства для хранения двоичной информации и методы её обработки наиболее просты и дёшевы. Поскольку в ЭВМ используется двоичная система счисления, необходимо переводить данные из десятичной формы

в двоичную (при вводе) и наоборот (при выводе результатов). Однако такой перевод легко автоматизируется, и многие пользователи даже не знают об этих внутренних преобразованиях.

В первых машинах использовались только числовые данные. В дальнейшем ЭВМ стали обрабатывать и другие виды информации (текст, графика, звук, видео), но это не привело к отмене принципа двоичного кодирования. Даже цифровые сигнальные процессоры[1], предназначенные для обработки цифровых сигналов в реальном времени, используют двоичное представление данных.

В истории известен пример успешной реализации *троичной ЭВМ «Сетунь»* (1959 г., руководитель проекта *Н. П. Брусенцов*), но он так и остался оригинальным эпизодом и не оказал влияния на эволюцию вычислительной техники. В первую очередь это связано с серьёзными проблемами, которые возникают при изготовлении элементов троичного компьютера на основе полупроводниковых технологий. Эти проблемы так и не были решены, тогда как наладить массовое производство аналогичных устройств для двоичных компьютеров оказалось значительно проще.

Принципы организации памяти

Принцип адресности памяти. Оперативная память машины состоит из отдельных **битов**. Для записи или считывания группы соседних битов объединяется в **ячейки памяти**, каждая из которых имеет свой адрес (номер). Нумерацию ячеек принято начинать с нуля.

Адрес ячейки памяти — это её номер.



Ячейка содержит минимально возможный считываемый из памяти объём данных: невозможно прочитать меньшее количество битов, а тем более отдельный бит.

Использование чисел для нахождения в памяти требуемых ячеек выглядит абсолютно естественно: в компьютерах любая информация кодируется числами, так что адреса ячеек — не исключение из этого фундаментального правила. Если номера соседних

[1] В англоязычной литературе их называют **DSP** — *Digital Signal Processor*.

ячеек отличаются на единицу, удобно организовывать их последовательную обработку.

Разрядность ячеек памяти (количество битов в ячейке) в разных поколениях была различной. Первоначально ЭВМ были построены исключительно для математических расчётов. Числа желательно было представлять как можно точнее, поэтому ячейки ОЗУ в первых машинах были длинными. Кроме чисел машина должна была хранить в памяти ещё и команды программы; как правило, в то время размер числовой ячейки совпадал с размером команды, что существенно упрощало устройство памяти.

Примерно на стыке второго и третьего поколений ЭВМ стали использовать для обработки символьной информации, что привело к серьёзному неудобству: в существующую числовую ячейку памяти помещалось 4–5 символов. Инженеры выбрали наиболее простое решение проблемы — уменьшить размер ячейки так, чтобы можно было обращаться к каждому символу отдельно. **Байтовая память**, основой которой стала **восьмибитная ячейка**, прекрасно зарекомендовала себя и используется в компьютерной технике до настоящего времени.

В результате перехода к «коротким» ячейкам памяти числа стали занимать несколько ячеек (байтов), каждая из которых имеет собственный адрес. На рисунке 5.8, *а* показана организация ячеек памяти первых ЭВМ, а на рис. 5.8, *б* — современная (байтовая) структура памяти.

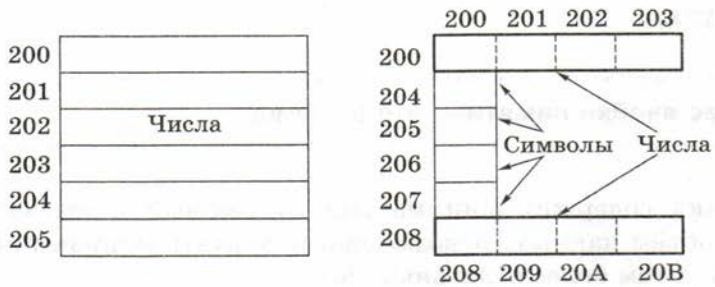


Рис. 5.8

На рисунке 5.8, *а* числа занимают по одной ячейке, причём номера этих ячеек отличаются на единицу. Справа показаны два 32-битных числа, которые хранятся в байтах 200–203 и 208–20B (адреса указаны в шестнадцатеричной системе). По принятому правилу за адрес числа принимается наименьший из адресов, так

что в данном случае адреса чисел — 200 и 208. Кроме того, на рис. 5.8, *б* между числами (в байтах с 204 по 207) размещены четыре однобайтных символа. Заметим, что современные компьютеры могут извлекать из памяти до восьми соседних байтовых ячеек за одно обращение к памяти.

Очень важно, что информация может считываться из ячеек и записываться в них в произвольном порядке, поэтому организованную таким образом память принято называть **памятью с произвольным доступом** (англ. **RAM** — *random access memory*). Чтобы лучше понять смысл этого термина, сравните такую память с магнитной лентой, данные с которой можно получить только путем последовательного чтения.

Часто термин RAM отождествляют с русским термином **ОЗУ — оперативное запоминающее устройство**. Это не совсем точно. Дело в том, что кроме ОЗУ существует еще одна разновидность памяти с произвольным доступом — **постоянное запоминающее устройство (ПЗУ**, англ. **ROM** — *Read Only Memory* — память только для чтения). Главное отличие ПЗУ от ОЗУ заключается в том, что при решении задач пользователя содержимое ПЗУ не может быть изменено. ПЗУ гораздо меньше ОЗУ по объёму, но это очень важная часть компьютера, поскольку в нём хранится доступное в любой момент программное обеспечение. Благодаря этому ПО компьютер сохраняет работоспособность даже тогда, когда в ОЗУ нет никакой программы.

Таким образом, ОЗУ и ПЗУ — это два вида памяти с произвольным доступом, обращение к данным в которых построено на основе принципа адресности.

Принцип иерархической организации памяти. К памяти компьютера предъявляются два противоречивых требования: её объём должен быть как можно больше, а скорость работы — как можно выше. Ни одно реальное устройство не может удовлетворить им одновременно. Любое существенное увеличение объёма памяти неизбежно приводит к уменьшению скорости её работы. Действительно, если память большая, то обязательно усложняется поиск в ней требуемых данных[1], а это сразу замедляет чтение

[1] Например, память большого объёма требует многоразрядного адреса, что, в свою очередь, приводит к очень большому количеству линий связи. В итоге приходится как-то изменять способ адресации, например передавать адрес по частям.

из памяти. Кроме того, чем быстрее работает память, тем она дороже, и, следовательно, меньше памяти можно установить за приемлемую для потребителей стоимость.

Чтобы преодолеть противоречие между объёмом памяти и её быстродействием, используют несколько различных видов памяти, связанных друг с другом. Когда в 1946 г. впервые формулировался этот принцип, в состав ЭВМ предполагалось включить всего два вида памяти: оперативную память и память на магнитной проволоке (предшественник устройств хранения данных на магнитной ленте). Дальнейшее развитие вычислительной техники подтвердило необходимость построения иерархической памяти: в современном компьютере уровней иерархии гораздо больше.

Выполнение программы

Принцип хранимой программы. Первые ЭВМ программировались путём установки перемычек на специальных панелях (рис. 5.9), так что процесс подготовки к решению задачи мог растянуться на несколько дней. Такое положение дел никого не устраивало, и в фон-неймановской архитектуре было предложено представлять команды в виде двоичного кода. Код программы, записанный заранее[1] на перфокарты или магнитную ленту, можно было ввести в машину достаточно быстро.

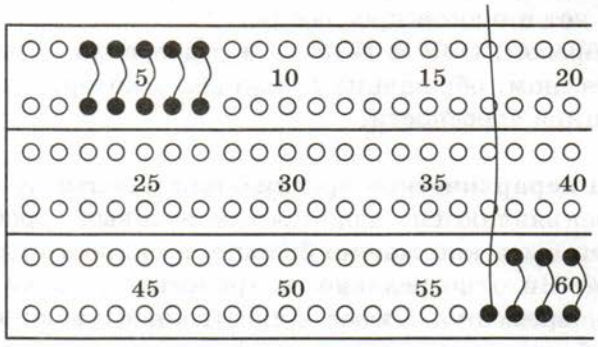


Рис. 5.9. Фрагмент коммутационной панели устройства IBM-557; требуемая операция получается соединением (коммутацией) соответствующих отверстий

[1] До появления персональных компьютеров для этого использовались специальные *устройства подготовки данных*. Такая схема ускоряла процесс ввода и исключала простои ЭВМ, связанные с длительным набором программ.

Поскольку команды программы и данные по форме представления стали одинаковыми, их можно хранить *в единой памяти*[1] вместе с данными. Не существует принципиальной разницы между двоичными кодами машинной команды, числа, символа и т. д. Это утверждение иногда называют **принципом однородности памяти**. Из него следует, что команды одной программы могут быть получены как результат работы другой программы. Именно так текст программы на языке высокого уровня переводится (транслируется) в машинные коды конкретной машины.

Код программы может сохраняться во внешней памяти (например, на дисках) и затем загружаться в оперативную память для повторных вычислений. Благодаря простоте замены программ, ЭВМ стали **универсальными устройствами**, способными решать самые разнообразные задачи в произвольном порядке и даже одновременно.

Принцип программного управления. Любая обработка данных в вычислительной машине происходит по программе. Принцип программного управления определяет наиболее общий механизм автоматического выполнения программы.

Важным элементом устройства управления в машине фон-неймановской архитектуры является специальный регистр — **счётчик адреса команд**[2]. В нём в любой момент хранится адрес команды программы, которая будет выполнена следующей.

Используя значение из счётчика, процессор считывает из памяти очередную команду программы, расшифровывает её и выполняет. Затем те же действия повторяются для следующей команды и т. д. Процессор выполняет команды по следующему алгоритму (его часто называют **основным алгоритмом работы процессора**):

1) из ячейки памяти, адрес которой записан в счётчике адреса команд, выбирается очередная команда программы; на время выполнения она сохраняется в специальном **регистре команд**;

---

[1] Известна также так называемая **гарвардская архитектура**, в которой программы и данные хранятся в разных областях памяти. Несмотря на повышение надёжности, она не получила широкого распространения.

[2] В различных процессорах этот регистр может называться по-разному: например, в семействе Intel он обозначается *IP* — Instruction Pointer (указатель на инструкцию).

2) значение счётчика адреса команд увеличивается так, чтобы он указывал на следующую команду;
3) выбранная команда выполняется (например, при сложении двух чисел оба слагаемых считываются в АЛУ, складываются и результат операции сохраняется в регистре или ячейке памяти);
4) далее весь цикл повторяется.

Таким образом, автоматически выполняя одну команду программы за другой, компьютер может исполнить любой линейный алгоритм. Для того чтобы в программе можно было использовать ветвления и циклы, необходимо нарушить естественную последовательность выполнения команд. Для этого существуют специальные **команды перехода**, которые на этапе 3 заносят в счётчик адреса новое значение — адрес перехода. Чаще всего в программах используется **условный переход**, т. е. переход происходит только при выполнении определённого условия.

Легко понять, что для запуска основного алгоритма работы процессора в счётчик адреса команд должно быть предварительно занесено начальное значение — адрес первой выполняемой команды. В первых ЭВМ оператор вводил этот адрес вручную. В современных компьютерах при включении питания в счётчик аппаратно заносится некоторое значение, которое указывает на начало программы, хранящейся в ПЗУ. Эта программа тестирует устройства компьютера и приводит их в рабочее состояние, а затем загружает в ОЗУ **начальный загрузчик операционной системы** (как правило, с диска). Ему и передаётся дальнейшее управление, а стартовая программа из ПЗУ завершает свою работу. Начиная с этого момента, поведение компьютера уже определяется установленным на нём программным обеспечением (см. главу 6).

Чтобы ускорить выполнение программы, основной алгоритм работы процессора был значительно усовершенствован. Идея была заимствована из **конвейерного** производства, где несколько рабочих одновременно выполняют различные операции (каждый над своим экземпляром изделия). Аналогично в современных микропроцессорах для каждого этапа выполнения команды создан отдельный аппаратный блок. Выполнив свою операцию, он передаёт результаты следующему блоку, а сам начинает выполнять очередную команду.

Проще всего понять этот механизм на примере первого этапа — выборки команды из ОЗУ. Специализированный блок

выборки извлекает из памяти последовательно расположенные команды, не дожидаясь окончания их обработки. Прочитанные команды размещаются в специальной рабочей памяти внутри микропроцессора. В итоге, когда первая из выбранных команд будет завершена, за следующей не придётся обращаться к ОЗУ, так как она уже находится во внутренней памяти микропроцессора. Учитывая, что обращение к ОЗУ занимает значительно большее времени, чем пересылка данных внутри процессора, такая *опережающая выборка* значительно ускоряет выполнение программы.

На практике применение конвейерного метода не так просто. Например, следующую команду часто не удаётся выполнить, поскольку она использует результат предыдущей, или сразу нескольким командам потребуется одновременно обратиться к ОЗУ. Тем не менее этот метод широко применяется в микропроцессорах. В некоторых моделях используются **параллельные конвейеры**, так что в некоторых случаях к моменту завершения выполнения одной команды уже готов результат следующей.

Что называют архитектурой

Описанные фон Нейманом и его соавторами классические принципы построения вычислительных устройств применялись во всех поколениях ЭВМ. В дополнение к ним в каждом конкретном семействе (PDP, ЕС ЭВМ, Apple, IBM PC и др.) формулируются свои собственные принципы устройства, благодаря которым обеспечивается аппаратная и программная совместимость моделей. Для пользователей это означает, что все существующие программы будут работать и на новых моделях того же семейства компьютеров. В литературе общие принципы построения конкретного семейства компьютеров называют **архитектурой**. К архитектуре обычно относят:

- принципы построения системы команд и их кодирования;
- форматы данных и особенности их машинного представления;
- алгоритм выполнения команд программы;
- способы доступа к памяти и внешним устройствам;
- возможности изменения конфигурации оборудования.

Стоит обратить внимание на то, что архитектура описывает именно общее устройство вычислительной машины, а не особенности изготовления конкретного компьютера (набор микросхем, тип жёсткого диска, ёмкость памяти, тактовая частота). Напри-

мер, наличие видеокарты как устройства для организации вывода информации на монитор входит в круг вопросов архитектуры. А вот является ли видеокарта частью основной платы компьютера или устанавливается на неё в виде отдельной платы, с точки зрения архитектуры значения не имеет. Иначе могло бы получиться, что для интегрированной в плату видеокарты потребовалась бы отдельная версия графического редактора!

Вопросы и задания

1. Найдите материалы, подтверждающие, что Джон фон Нейман не был единоличным автором «фон-неймановской» архитектуры ЭВМ.
2. Перечислите принципы фон-неймановской архитектуры и кратко объясните каждый из них.
3. Назовите основные компоненты вычислительного устройства. Каково их назначение? Согласны ли вы с тем, что полученный набор узлов логичен и обоснован?
4. В чём состоит принцип двоичного кодирования?
5. Вспомните, как кодируются в компьютере числа, тексты, графика. Соблюдается ли принцип двоичного кодирования?
6. По какому алгоритму вводимые в компьютер десятичные числа можно перевести во внутреннее двоичное представление? Как перевести обратно результаты расчёта?
7. Что такое ячейка памяти? Что такое адрес ячейки?
8. Что вы знаете о разрядности ячеек ОЗУ разных поколений?
9. Почему появилась байтовая память?
10. Можно ли заменить в ячейке памяти содержимое одного бита, не затрагивая значений соседних? Почему?
11. Приведите примеры различных типов данных и назовите их разрядность. Сколько байтов памяти потребуется для хранения данных каждого из этих типов?
12. Что такое иерархическая организация памяти?
13. Почему большая по объёму память обычно работает медленнее, чем маленькая?
14. В чём состоит принцип хранимой программы?
15. Где может храниться программа?
16. Можно ли к нечисловым данным (символам, графическим и звуковым данным) применять арифметические операции?
17. Как вы понимаете фразу «Любая обработка данных в вычислительной машине происходит по программе»? Чем компьютер в этом отношении отличается от простого калькулятора?
18. Сформулируйте основной алгоритм выполнения команды в компьютере.

19. Что такое счётчик адреса команд и какова его роль в основном алгоритме?
20. Опишите, что происходит в момент включения компьютера с точки зрения принципа программного управления.
21. Можно ли нарушить последовательность выполнения команд в программе? Для чего это может потребоваться?
22. Всегда ли в новом компьютере есть какая-либо программа?
23. Что такое конвейер и как он работает при выполнении программы?
*24. Почему команды перехода нарушают работу конвейера?
25. Какие из принципов, предложенных в работе «Предварительное рассмотрение логической конструкции электронного вычислительного устройства», продолжают применяться в современных компьютерах безо всяких изменений, а какие сохранились, но в несколько изменённом виде? Объясните, почему потребовались эти изменения.
26. Что такое архитектура? Какие детали устройства компьютера к ней не относятся?
27. В чём преимущества единой архитектуры семейств ЭВМ для пользователей и для производителей?
28. Какие семейства вычислительных машин вы знаете?

Подготовьте сообщение

а) «Джон фон Нейман и его вклад в науку»
б) «Троичная ЭВС "Сетунь"»
в) «Гарвардская архитектура»
г) «Архитектуры современных компьютеров»

Задачи

*1. Используя условные команды «считать байт из памяти», «записать байт в память», а также битовые логические операции «И», «ИЛИ» и «НЕ», составьте алгоритм, который в байте, хранящемся в памяти:
   а) установит младший бит данных в единицу, не затрагивая содержимого остальных двоичных разрядов;
   б) сбросит его в ноль.
2. Описанная в «Предварительном рассмотрении...» конструкция ЭВМ имела ячейки памяти, состоящие из 40 двоичных разрядов. Сколько современных байтовых ячеек потребуется для хранения числа такой же разрядности?
3. В языке Паскаль есть тип чисел word, значением которого являются целые положительные 16-разрядные числа. Сколько байт занимает такое число в памяти? Какое максимальное значение может иметь этот тип чисел?

4. В микропроцессорах семейства Intel для увеличения на единицу одного из регистров процессора используется команда, имеющая код 41_{16}. Пользуясь таблицей символов, определите, какая буква соответствует этому же самому коду. Если рассматривать этот код как целое число, чему оно будет равно в десятичной системе счисления?

*5. Изучите содержимое текстового файла, используя программу, которая способна отображать данные в виде шестнадцатеричных кодов. Попробуйте найти коды известных вам символов. Сделайте то же самое с графическим файлом: удастся ли вам найти там те же самые коды («символы»)?

*6. Найдите таблицу кодов команд процессора Intel. Сравните эти коды с кодами, которые вы нашли в предыдущем задании. Удалось ли найти совпадения?

§ 33 Магистрально-модульная организация компьютера

Что понимается под устройством компьютера?

Компьютер — это пример очень сложной техники. При изучении таких систем возможно несколько разных подходов. Например, можно изучать:

- устройство конкретного экземпляра компьютера: набор микросхем, тип основной платы, конструкцию и разновидности модулей памяти и т. п.;
- семейство компьютеров, например IBM-совместимых персональных компьютеров;
- различные конструкции компьютеров (настольные компьютеры, портативные компьютеры, карманные компьютеры);
- функциональное устройство компьютера, т. е. его основные узлы и способы взаимодействия между ними.

Каждый из этих подходов полезен при решении определённых задач. Так для настройки конкретного компьютера необходимо точно знать марки и параметры его устройств. Определить эти

данные можно с помощью специального программного обеспечения. К сожалению, любые знания в этой области очень быстро устаревают, поскольку аппаратура постоянно меняется.

Если изучать особенности одного семейства компьютеров, мы получим «однобокое» представление об устройстве компьютерной техники, так как каждое семейство имеет свои особенности.

Современные компьютеры очень разнообразны и поэтому имеют самую различную конструкцию и внешний вид. **Настольный ПК** состоит из системного блока и подключённых к нему внешних устройств. Такая конструкция удобна для пользователя, поскольку все устройства можно разместить на столе так, как ему хочется.

В **переносных компьютерах** весь минимально необходимый набор устройств собран в одном корпусе. Сейчас такие компьютеры называют **ноутбуками** (англ. *notebook* — тетрадь, блокнот). По своим вычислительным возможностям они практически не уступают настольным ПК.

Растёт популярность так называемых **нетбуков** (от слов «Интернет» и «ноутбук») — так называют очень маленькие и лёгкие переносные компьютеры. Кроме меньшего размера и веса нетбуки отличаются от ноутбуков бóльшим временем автономной работы и меньшей стоимостью. Нетбуки предназначены для пользователей, которые применяют компьютер главным образом для работы в Интернете и подготовки простых документов. Их используют люди, совершающие большое число поездок.

Карманные персональные компьютеры (КПК) умещаются на ладони. Их называют также **наладонниками** (англ. *palmtop*) и **PDA** (англ. *Personal Digital Assistant* — персональный цифровой помощник). У большинства из них даже нет клавиатуры, а для ввода данных нажимают пластиковой палочкой (она называется **стилусом**) или пальцем на **сенсорный** (реагирующий на прикосновение) **экран**.

Вместе с тем мощные серверы и суперкомпьютеры по-прежнему собираются в виде крупных «шкафов», напоминающих ЭВМ предыдущих поколений. Наконец, нельзя не упомянуть и о бытовой электронике, которая всё больше и больше приближается к традиционным компьютерам.

Разнообразие типов современных компьютеров говорит о том, что конструкция — это не самое главное. В то же время, как показано в § 32, их функциональное устройство практически не изменяется. Поэтому далее мы подробно рассмотрим основные узлы

компьютера (процессор, память и устройства ввода и вывода) и взаимодействие между ними.

Взаимодействие устройств

Процессор должен обмениваться данными с внутренней памятью и устройствами ввода и вывода. Выделить отдельные каналы для связи процессора с каждым из многочисленных устройств нереально. Вместо этого сделана общая линия связи, доступ к которой имеют все устройства, использующие её по очереди. Такой информационный канал называется шиной.

Шина (или **магистраль**) — это группа линий связи для обмена данными между несколькими устройствами компьютера.

Традиционно шина делится на три части (рис. 5.10) — это:

- **шина данных**, по которой передаются данные;
- **шина адреса**, определяющая, куда именно передаётся информация;
- **шина управления**, которая организует процесс обмена (несёт сигналы чтение/запись, обращение к внутренней/внешней памяти, данные готовы/не готовы и т. п.).

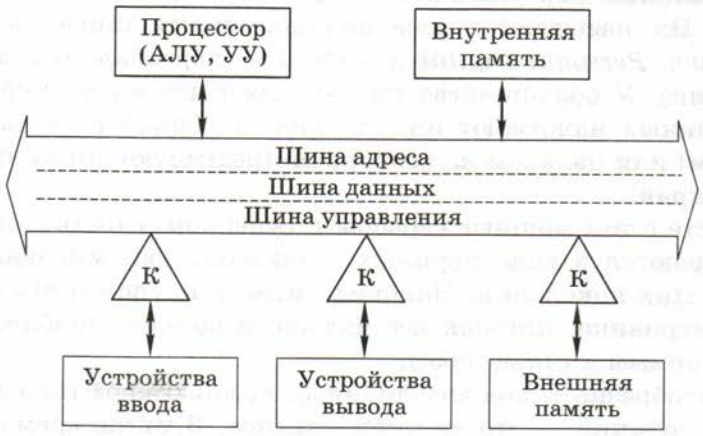


Рис. 5.10

Рассмотрим процесс записи данных из процессора в память. На шину данных процессор выставляет данные для записи, на шину адреса — нужный адрес памяти, а на шину управления — сигналы для записи информации в память. Далее он вынужден ожидать, пока данные будут «взяты» с шины. В это время все остальные устройства постоянно «слушают» шину (проверяют её состояние). В нашем примере по сигналам на шине память обнаруживает, что для неё имеются данные. Она сохраняет их по заданному адресу и должна по шине управления сообщить процессору, что операция завершена. На практике, учитывая высокую надёжность работы памяти, сигнал подтверждения часто не используется: процессор просто выжидает определённое время и продолжает выполнение программы. Из этого примера понятно, что для успешного обмена данными по шине должны быть введены чёткие правила (их принято называть **протоколом шины**), которые должны соблюдать все устройства.

По сравнению с первыми ЭВМ, взаимодействие процессора с внешними устройствами организуется теперь по-другому. В классической архитектуре процессор контролировал все процессы ввода/вывода. Получалось так, что быстродействующий процессор тратил много времени на ожидание при работе с значительно более медленными внешними устройствами. Поэтому появились специальные электронные схемы, которые руководят обменом данными между процессором и внешними устройствами. В третьем поколении такие устройства назывались **каналами ввода/вывода**, а в четвёртом — **контроллерами**[1] (на рис. 5.10 они обозначены буквой К).

Контроллер — это электронная схема для управления внешним устройством и простейшей предварительной обработки данных.

Современный контроллер — это специальный микропроцессор, предназначенный для обслуживания одного или нескольких однотипных устройства ввода/вывода (УВВ) или внешней памяти. Нагрузка на центральный процессор при этом существенно снижается, и это увеличивает эффективность работы всей системы

[1] Это название происходит от английского слова *control* — управление; не следует путать с русским словом «контролёр».

в целом. Контроллер, собранный в виде отдельной микросхемы, называют **микроконтроллером**.

В качестве примера рассмотрим контроллер современного жёсткого диска. Его основная задача — по принятым от процессора координатам найти на диске требуемые данные, прочитать их и передать в ОЗУ. Но контроллер способен выполнять и другие, порой весьма нетривиальные функции. Так, он сохраняет в служебной области диска информацию обо всех имеющихся на магнитной поверхности некачественно изготовленных секторах (а их при современной высокой плотности записи избежать не удаётся!) и способен «на ходу» подменять их резервными, что создаёт видимость диска, который полностью свободен от дефектов.

Как видно из приведённой на рис. 5.10 схемы, теперь данные могут передаваться между внешними устройствами и ОЗУ напрямую, минуя процессор. Кроме того, наличие шины существенно упрощает подсоединение к ней новых устройств. Архитектуру, которую можно легко расширять за счёт подключения к шине новых устройств, часто называют **магистрально-модульной архитектурой**.

Если спецификация на шину (детальное описание всех её логических и физических параметров) является открытой (опубликована), то производители могут разрабатывать к такой шине любые дополнительные устройства. Такой подход называют **принципом открытой архитектуры**. При этом в компьютере предусмотрены стандартные разъёмы для подключения новых устройств, удовлетворяющих стандарту. Поэтому пользователь может собрать такой компьютер, который ему нужен. Необходимо только помнить, что при подключении любого нового устройства нужно установить специальную программу — **драйвер**, которая управляет обменом данными между этим устройством и процессором.

В современных компьютерах для повышения эффективности работы используется несколько шин, например одна — между процессором и памятью, другая связывает процессор с видеосистемой и т. д.

Обмен данными с внешними устройствами

Существуют три режима обмена данными между центральным процессором (ЦП) и внешними устройствами:

- программно управляемый ввод/вывод;
- обмен с устройствами по прерываниям;
- прямой доступ к памяти (ПДП).

При **программно управляемом обмене** все действия по вводу или выводу предусмотрены в теле программы. Процессор полностью руководит ходом обмена, включая ожидание готовности периферийного устройства и прочие временны́е задержки, связанные с процессами ввода/вывода. *Достоинства* этого метода — простота и отсутствие дополнительного оборудования, *недостаток* — большие потери времени из-за ожидания быстро работающим процессором более медленных устройств ввода/вывода.

При **обмене по прерываниям** устройства ввода/вывода в случае необходимости сами «требуют внимания» процессора. Например, клавиатура оповещает процессор каждый раз, когда была нажата или отпущена клавиша; всё остальное время процессор выполняет программу, «не отвлекаясь» на клавиатуру. Когда прерывание произошло, ЦП «откладывает» на некоторое время выполнение основной программы и переходит на служебную программу обработки прерывания. Завершив его обработку, ЦП снова возвращается к тому месту программы, где она оказалась прервана. При этом основная программа даже «не заметит» возникшей задержки. Этот режим обмена более сложен, но зато значительно эффективнее — процессор не тратит время на ожидание.

Представим себе, что в кабинете начальника идёт совещание и в этот момент по телефону поступает важная информация, требующая немедленного принятия решения. Секретарша, не дожидаясь конца совещания, сообщает начальнику о звонке. Тот, прервав свое выступление, снимает трубку, выясняет суть дела и сообщает свое решение. Затем он продолжает совещание, как ни в чем не бывало. Здесь роль ЦП играет начальник, а телефонный звонок — это запрос (требование) на прерывание. «Секретарша» в компьютере тоже предусмотрена — это контроллер прерываний, анализирующий и сортирующий все поступающие прерывания с учётом их важности (*приоритета*).

Механизм прерываний используется не только в аппаратной части, но и в программах, которые основаны на обработке событий (нажатий на клавиши, команд управления от мыши и т. п.). Такая технология лежит в основе современных операционных систем и применяется в системах разработки программ Microsoft Visual Studio, Delphi, Lazarus и им подобных.

В обоих описанных выше вариантах управление обменом выполнял центральный процессор. Именно он извлекал из памяти

выводимые данные (или записывал туда вводимые), подсчитывал их количество и полностью контролировал работу шины. Если передаваемые данные не требуют сложной обработки, ЦП напрасно расходует время на проведение обмена. Чтобы освободить процессор от этой работы и увеличить скорость передачи крупных блоков данных от устройства ввода в память и обратно, применяется **прямой доступ к памяти (ПДП**, англ. **DMA** — *Direct Memory Access*).

Принципиальное отличие ПДП состоит в том, что в этом режиме процессор *не производит обмен*, а только подготавливает его, программируя контроллер ПДП: устанавливает режим обмена, а также передаёт начальный адрес ОЗУ и количество циклов обмена. Далее контроллер в ходе ПДП самостоятельно наращивает первое значение и уменьшает второе, что позволяет освободить центральный процессор.

Изложенный материал о режимах ввода/вывода может быть сведён в табл. 5.1 (здесь УВВ обозначает устройство ввода/вывода)

Таблица 5.1

Вид обмена	Начинает обмен	Руководит обменом	Текущая программа	Программа обмена
Программный	ЦП	ЦП	Программа обмена — часть текущей программы	
Прерывания	УВВ	ЦП	Прерывается	Специальная подпрограмма
ПДП	УВВ, ЦП	Контроллер ПДП	Выполняется параллельно	Отсутствует (обмен идёт аппаратно)

Вопросы и задания

1. Как можно определить, какие именно платы и устройства установлены в вашем компьютере? Для чего это может потребоваться?
2. Как вы думаете, что более полезно для глубокого понимания работы компьютера: изучение функционального устройства компьютера или изучение его конструкции?
3. Как устройства компьютера обмениваются данными?
4. Что такое шина? Почему обмен данными между устройствами компьютера с помощью шины оказался наилучшим решением?

5. Из каких частей состоит шина? Охарактеризуйте каждую из них.
6. Что такое магистрально-модульная архитектура и в чём её главное достоинство?
7. В чём заключается принцип открытой архитектуры?
8. Используя приведённое в параграфе объяснение процесса записи данных в память, попробуйте объяснить, как происходит считывание данных из ячейки памяти с заданным адресом.
9. Что такое контроллер и для чего он нужен?
10. Объясните, как использование контроллеров позволяет повысить быстродействие компьютера в целом.
11. Сравните магистрально-модульную архитектуру компьютера с классической. Выделите наиболее перегруженный блок на каждой из схем.
12. Почему в современном компьютере несколько шин?
13. Что требуется для успешного присоединения к компьютеру нового устройства?
14. Как расшифровывается сокращение ПДП и что это такое?
15. Как выполняется обмен данными в режиме ПДП?
16. Расскажите о разных режимах обмена данными с внешними устройствами.
17. Предложите наиболее подходящий режим обмена данными с клавиатурой.
18. Какой режим лучше всего подходит для обмена данными с жёстким диском?
19. Где в программировании применяются принципы обработки прерываний? Приведите примеры.

Подготовьте сообщение

а) «Принцип открытой архитектуры»
б) «Что такое прерывание?»
в) «Зачем нужны контроллеры?»
г) «Прямой доступ к памяти»

§ 34 Процессор

Центральным устройством, во многом определяющим возможности компьютера, является процессор (рис. 5.11).

Процессор — это устройство, предназначенное для автоматического считывания команд программы, их расшифровки и выполнения.

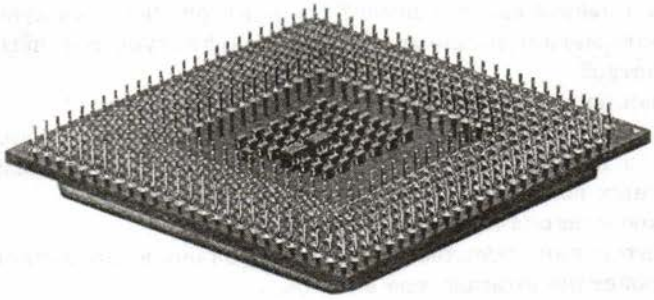


Рис. 5.11. Вид микропроцессора со стороны выводов

Название «процессор» происходит от английского глагола «to process» — обрабатывать. Иными словами, процессор — это блок компьютера, который автоматически обрабатывает данные по заданной программе[1].

Процессор, изготовленный в виде большой или сверхбольшой интегральной схемы (БИС, СБИС), называется **микропроцессором**.

Любой процессор обязательно включает в себя две важные части, каждая из которых решает свои задачи:

- **арифметико-логическое устройство (АЛУ)**, выполняющее обработку данных;
- **устройство управления (УУ)**, которое управляет выполнением программы и обеспечивает согласованную работу всех узлов компьютера.

Арифметико-логическое устройство

В простейшем случае АЛУ состоит из двух регистров, сумматора и схем управления операциями (об устройстве сумматора и регистров можно прочитать в § 24). При выполнении операций в регистры помещаются исходные данные, а в сумматоре они складываются.

Как показано в главе 4, все остальные арифметические операции могут быть тем или иным способом сведены к сложению. Тем не менее нередко для ускорения умножения и деления инженеры

---

[1] Не следует путать процессор как аппаратный блок с мощными программами обработки данных, которые также часто называют процессорами (например, текстовый процессор или табличный процессор).

идут на усложнение АЛУ. Например, в процессорах широко используется метод умножения чисел с использованием таблиц, в которых записаны готовые произведения небольших чисел.

АЛУ не только выполняет вычисления, но и анализирует полученный результат. Обычно проверяются два свойства: равенство нулю (совпадение всех разрядов сумматора с нулем) и отрицательность результата (определяется проверкой знакового разряда — см. § 27). Результаты этого анализа заносятся в определённые биты **регистра состояния** процессора. Используя эти значения, можно сделать вывод об истинности или ложности условий $R = 0$, $R \neq 0$, $R > 0$, $R < 0$, $R \geq 0$, $R \leq 0$, где R обозначает результат операции. Это позволяет организовать ветвления в программе, например для неотрицательного числа вычислять квадратный корень, а иначе — выдать сообщение об ошибке.

Как правило, АЛУ работает только с целыми числами. Операции с вещественными числами выполняются в **математическом сопроцессоре**, который встроен внутрь современных микропроцессоров.

Устройство управления

Главная задача устройства управления — обеспечить автоматическое выполнение последовательности команд программы в соответствии с основным алгоритмом работы процессора (см. § 32). Устройство управления выполняет следующие действия:

- извлечение из памяти очередной команды;
- расшифровка команды, определение необходимых действий;
- определение адресов ячеек памяти, где находятся исходные данные;
- занесение в АЛУ исходных данных;
- управление выполнением операции;
- сохранение результата.

Таким образом, выполнение каждой **машинной команды** состоит из элементарных действий, которые называются **микрокомандами**.

В зависимости от сложности, машинная команда может быть выполнена за различное число микрокоманд. Например, пересылка числа из одного внутреннего регистра микропроцессора в другой требует значительно меньшего числа действий, чем умно-

жение. Команды, работающие с оперативной памятью, выполняются дольше, чем команды, работающие только с регистрами процессора.

Каждая из микрокоманд машинной команды запускается с помощью управляющего импульса. Опорную последовательность импульсов для этих целей УУ получает от **генератора тактовых импульсов**. Интервал между двумя соседними импульсами называется **тактом**.

Если две микрокоманды полностью независимы друг от друга, то их можно выполнить одновременно (за один такт), даже если они принадлежат к разным командам программы. Такая оптимизация широко применяется в современных процессорах для организации **конвейерной обработки** ради увеличения быстродействия.

Регистры процессора

Кроме регистров АЛУ и УУ в микропроцессоре есть много других регистров. Большинство из них — внутренние, они недоступны программисту. Однако есть несколько регистров, специально предназначенных для использования программным обеспечением. Их часто называют **регистрами общего назначения (РОН)**, подчёркивая тем самым универсальность их функций. В РОН могут храниться не только сами данные (числа, коды символов и т. д.), но и адреса ячеек памяти, где эти данные находятся. Например, если требуется обработать последовательные ячейки памяти, то к содержимому такого регистра нужно каждый раз прибавлять размер ячейки.

Количество регистров и их устройство в разных процессорах отличается друг от друга. Например, в процессорах семейства *Intel* имеется небольшой набор 64-разрядных РОН. Ради обеспечения программной совместимости со старыми (32- и 16-разрядными) процессорами эти РОН имеют вложенную структуру, напоминающую матрёшку (рис. 5.12).

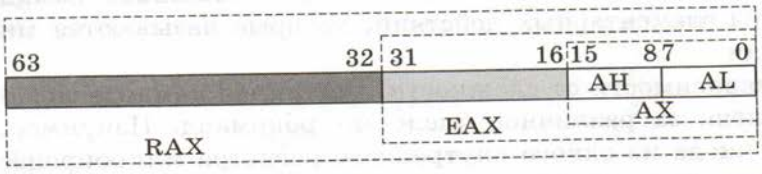


Рис. 5.12

На рисунке 5.12 показана структура 64-разрядного регистра RAX. Его младшие 32 бита (с нулевого по 31-й) образуют регистр EAX для 32-разрядных вычислений. К младшим 16 битам EAX (0–15), в свою очередь, можно также обращаться как к самостоятельному регистру AX. Наконец, биты 0–7 и 8–15 образуют два 8-разрядных регистра AL и AH. Отчётливо видно, что наращивание разрядности процессоров семейства Intel происходило постепенно. Такая структура регистров обеспечивает совместимость с предыдущими моделями и позволяет процессору легко обрабатывать 8-, 16-, 32- и 64-разрядные данные.

Кроме рассмотренного выше регистра RAX в процессорах Intel есть аналогичным образом устроенные регистры RBX, RCX и RDX, а также некоторые другие. Это регистры неравноценны, по справочникам можно определить, как и с каким регистром работает та или иная команда.

Основные характеристики процессора

Мы уже говорили о том, что для организации выполнения команд в компьютере есть **генератор тактовых импульсов**, каждый из которых «запускает» очередной **такт** машинной команды. Очевидно, что чем чаще следуют импульсы от генератора, тем быстрее будет выполняться операция. Следовательно, тактовая частота, измеряемая количеством тактовых импульсов в секунду, может быть характеристикой быстродействия процессора.

Тактовая частота — количество тактовых импульсов в одну секунду.



В настоящее время тактовая частота измеряется в гигагерцах, т. е. в миллиардах (10^9) импульсов в секунду. Эту частоту нельзя установить сколь угодно высокой, поскольку процессор может просто не успеть выполнить действие очередного такта до прихода следующего импульса.

Нужно понимать, что использовать тактовую частоту для сравнения быстродействия процессоров можно только в том случае, если оба процессора устроены одинаково. Например, если какая-то команда в одном из процессоров выполняется за два такта, а в другом — за три, то при равенстве частот первый будет работать в полтора раза быстрее.

Приближённо можно считать, что процессор выполняет за один такт одну простую команду (типа пересылки числа из регистра в регистр). Тогда при тактовой частоте 4 ГГц за одну секунду выполняется около 4 миллиарда таких операций. Это примерная оценка, потому что при конвейерном методе скорость выполнения команд сильно зависит от множества факторов, например от порядка следования команд в программе.

Другая характеристика, позволяющая судить о производительности процессора, — это его разрядность.



Разрядность — это максимальное количество двоичных разрядов, которые процессор способен обработать за одну команду.

Чаще всего разрядность определяют как размер регистров процессора в битах.

Однако важны также разрядности шины данных и шины адреса, которые поддерживает процессор. **Разрядность шины данных** — это максимальное количество битов, которое может быть считано за одно обращение к памяти. **Разрядность шины адреса** — это количество адресных линий; она определяет максимальный объём памяти, который способен поддерживать процессор. Этот объём памяти часто называют величиной **адресного пространства**, он вычисляется по формуле 2^R, где R — количество разрядов шины адреса.

Все три разрядности могут не совпадать. Так, у процессора *Pentium II* были 32-разрядные регистры, разрядность шины данных — 64 бита, а шины адреса — 36 битов.

Система команд процессора

Каждая модель процессора имеет собственную систему команд. Поэтому, как правило, процессоры могут выполнять только программы, написанные специально для них. Тем не менее обычно новые процессоры одной и той же серии (например, процессоры Intel) поддерживают все команды предыдущих моделей.

В системах команд разных процессоров есть много общего. Они обязательно включают следующие группы машинных команд:

- команды передачи (копирования) данных;
- арифметические операции;

- логические операции, например «НЕ», «И», «ИЛИ», «исключающее ИЛИ»;
- команды ввода и вывода;
- команды переходов.

Существуют два основных подхода к построению системы команд процессора:

- процессоры с полным набором команд (*англ.* CISC = Complex Instruction Set Computer);
- процессоры с сокращённым набором команд (*англ.* RISC = Reduced Instruction Set Computer).

CISC-процессоры содержат широкий набор разнообразных команд. При этом на скорость их выполнения обращают меньшее внимание, главное — удобство программирования. При разработке **RISC-процессоров** набор команд, наоборот, весьма ограничен, но это позволяет значительно ускорить их выполнение. Многие современные процессоры (например, процессоры Intel) — гибридные, у них полный набор команд, каждая из которых автоматически заменяется серией более простых, выполняемых RISC-ядром. Это позволяет совместить *достоинства* обоих подходов.

Почти все команды, входящие в систему команд компьютера, состоят из двух частей — операционной и адресной. **Операционная часть — код операции** — указывает, какое действие необходимо выполнить. **Адресная часть** описывает, где хранятся исходные данные и куда поместить результат. Часто исходные данные для команды (содержимое регистров или ячеек памяти, константы) называют **операндами**.

Рассмотрим для примера одну из наиболее простых команд процессора Intel, которая состоит из четырёх байтов и имеет шестнадцатеричный код 81 C2 01 01. Она может быть разбита на три неодинаковые по длине части:

- код операции 81C обозначает сложение содержимого регистра с константой;
- первый операнд 2 — это условное обозначение регистра DX;
- второй операнд — константа 0101, которая добавляется к содержимому регистра.

Отметим, что система команд процессоров Intel очень сложна и плохо подходит для изучения в школьном курсе информатики.

Вопросы и задания

1. Для чего нужен процессор? Почему он так называется?
2. Какие узлы входят в состав процессора? Зачем нужны АЛУ и УУ?
3. Как устроено АЛУ в простейшем случае? Как в АЛУ используется сумматор?
4. Почему удобно, что АЛУ автоматически сравнивает результат действия с нулем?
5. Подумайте, как с помощью логических операций с битами сумматора установить факт его равенства или неравенства нулю.
6. Для чего служит математический сопроцессор?
7. Какую роль играет УУ в автоматическом выполнении программ?
8. Как называется элементарное действие в машинной команде?
9. Зачем нужен генератор тактовых импульсов?
10. Что такое РОН? Для каких целей он может использоваться?
*11. Найдите в Интернете информацию о регистрах процессора Intel. Постарайтесь разобраться в назначении наиболее важных из них.
12. Что такое тактовая частота и как она влияет на быстродействие компьютера?
13. Тактовые частоты двух процессоров, изготовленных фирмами Intel и AMD, равны. Означает ли это, что их быстродействие одинаково? Обоснуйте свой вывод.
*14. Объясните, как применение конвейера влияет на количество команд, выполняемых за один такт.
15. На что влияет разрядность процессора? Какие разновидности разрядности вы знаете? Что характеризует каждая из них?
16. Какие группы операций входят в систему команд любого процессора?
17. Что такое RISC- и CISC-процессоры? Чем они различаются?
18. Какие части можно выделить в команде процессора?

Подготовьте сообщение

а) «CISC и RISC-процессоры»
б) «Процессоры Intel и AMD»
в) «Конвейерная обработка данных»
г) «Многоядерные процессоры»

Задачи

1. Обозначим символом Z бит, определяющий факт равенства результата R нулю, а символом N — бит, фиксирующий отрицательность R:
   - $Z = 1$ при $R = 0$, и $Z = 0$ в противном случае;
   - $N = 1$ при $R < 0$, и $N = 0$ в противном случае.

Напишите логическое выражение, включающее значения N и Z, которое даёт:
а) 1 при $R \le 0$, и 0 в противном случае;
б) 1 при $R > 0$, и 0 в противном случае.

2. Оцените, сколько миллиардов простых операций типа пересылки регистр—регистр может выполнить за одну минуту процессор с тактовой частотой 1 ГГц.
3. Сопоставьте тактовую частоту процессора с максимальной частотой звуковых колебаний, которые слышит человек. Что можно сказать о возможностях современного компьютера в обработке звуковой информации?
4. Какое максимальное десятичное целое число без знака можно поместить в 32-разрядный регистр?
5. Сколько символов, закодированных в двухбайтной кодировке UNICODE, можно загрузить одновременно в 64-разрядный регистр?
6. Процессор Pentium II имеет 36-разрядную шину адреса. Какой объём памяти он может адресовать?

§ 35
Память

Как мы уже знаем, процессор способен выполнять программу, но её команды хранятся в памяти. Таким образом, память — это другое устройство, без которого вычислительный автомат не может быть построен. Кроме того, память одновременно используется (что не менее важно) и для хранения обрабатываемых данных.

Память — это устройство компьютера, которое используется для записи, хранения и выдачи по запросу команд программы и данных.

Существует большое количество видов памяти, которые различаются по устройству, организации, функциям и т. д. Обычно выделяют **внутреннюю** и **внешнюю память**. Термины эти имеют историческое происхождение, связанное с конструкцией первых ЭВМ: одна часть памяти находилась внутри главного шкафа (в котором размещался процессор), а другая — вне его.

Современные компьютеры, конечно, выглядят совсем по-другому, из-за чего названия утратили свою прежнюю наглядность. Тем не менее деление памяти на два типа по-прежнему сохраняется. Различие между ними кроется, прежде всего, в назначении. Внутренняя память предназначена для хранения программ и данных, которые используются для задач, решаемых в данный момент. А внешняя память служит для того, чтобы сохранить данные на длительный срок, пока они не потребуются, именно поэтому её ещё часто называют **долговременной**.

Внутренняя память

Внутренняя память — часть памяти компьютера, которая используется для хранения программ и данных во время решения задачи.

Часто её называют *основной памятью*. В состав внутренней памяти входят **ОЗУ** (оперативное запоминающее устройство) и **ПЗУ** (постоянное запоминающее устройство).

Внутренняя память строится в соответствии с базовыми принципами, описанными ранее в § 32. Основное отличие внутренней памяти от внешней — произвольный доступ к отдельным ячейкам памяти по их адресам (обращение к внешней памяти происходит иначе, см. далее).

Информация, хранящаяся в ОЗУ, считается временной (оперативной), поэтому пользователь должен сам сохранять необходимые данные во внешней памяти.

Часто говорят, что при выключении питания информация в ОЗУ пропадает. Строго говоря, это не совсем правильно, поскольку существуют элементы памяти, способные сохранять свое состояние даже после отключения питания. Однако при повторном включении (или перезагрузке) компьютера программное обеспечение не способно восстановить, где и какая информация находилась «в прошлый раз». Именно поэтому, если при наборе текста перезагрузить компьютер, работу придётся повторять заново.

Внутренняя память может быть построена на основе самых разных технологий. Самые первые ЭВМ имели ОЗУ на электронно-лучевых трубках, причем их количество соответствовало разрядности памяти (каждый бит числа считывался из отдельной

трубки). Затем появилась память на магнитных сердечниках (рис. 5.13). Намагниченное состояние сердечника соответствовало единичному состоянию бита, ненамагниченное — нулевому. Заметим, что данные в магнитных ячейках памяти полностью сохранялись и после выключения питания. Наконец, развитие микроэлектроники позволило изготовить компактную полупроводниковую память (рис. 5.14), которая сейчас и применяется в персональных компьютерах.

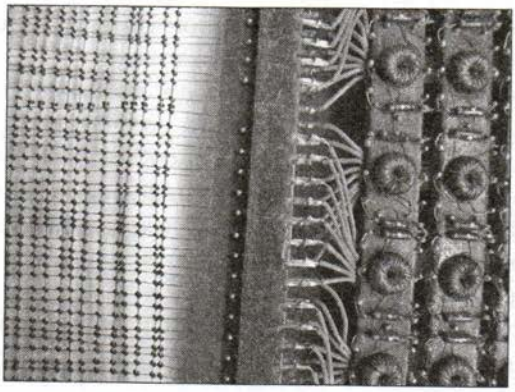


Рис. 5.13. Магнитное ОЗУ:
слева — биты памяти, справа — устройство выборки нужного адреса

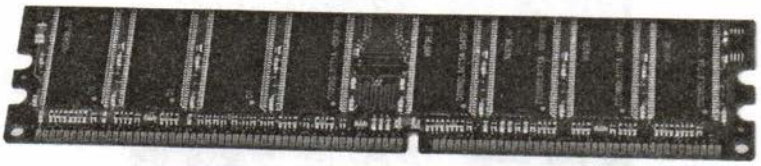


Рис. 5.14. Модуль полупроводниковой памяти

Существуют **два типа оперативной памяти**, отличающиеся по технологии изготовления, — **статическая** и **динамическая**. Первая строится *на триггерах* (об устройстве триггера см. в § 24), а вторая — *на полупроводниковых конденсаторах*. Конденсатор намного проще и меньше триггера, так что на одном и том же кристалле можно сделать гораздо больше запоминающих элементов динамического типа, чем статического. Поэтому динамическая память имеет бóльшую ёмкость и меньшую стоимость, чем статическая. К сожалению, у неё есть очень существенный недос-

таток: она работает намного медленнее статической. Сейчас в персональных компьютерах используется динамическая оперативная память.

Что касается ПЗУ, то технологии их изготовления также постепенно совершенствовались. Первоначально информация в ПЗУ заносилась только на заводе. Затем появились **программируемые ПЗУ**, которые потребитель мог заполнить сам, поместив «чистую» («пустую») микросхему в специальное устройство — **программатор**. В некоторых микросхемах этого типа в качестве запоминающих элементов использовали тонкие токопроводящие перемычки. Наличие перемычки означало единицу. Программатор мощными импульсами тока пережигал нужные перемычки, тем самым устанавливая биты в нулевое состояние[1]. Очевидно, что процесс записи информации таким способом был необратимым.

Позднее появились **перепрограммируемые ПЗУ**, в которых очистка информации сначала производилась ультрафиолетовыми лучами, а затем — с помощью электрических импульсов. Современные перепрограммируемые ПЗУ используют флэш-память (рис. 5.15). Каждый элемент такой памяти изготовлен на основе особой разновидности транзисторов, так что это тоже полупровод-

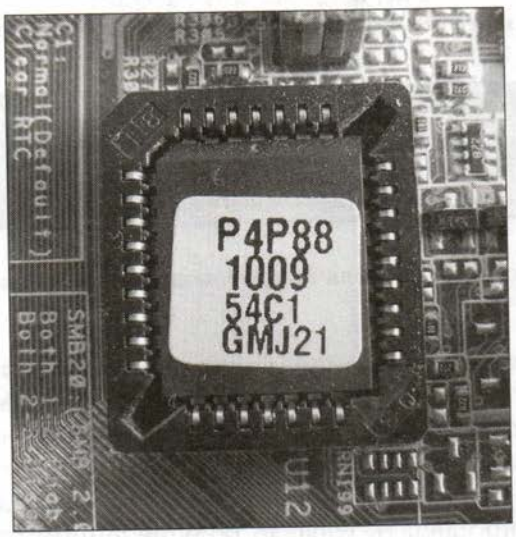


Рис. 5.15. Флэш-BIOS на плате компьютера

[1] Так сгорают плавкие предохранители в бытовой аппаратуре.

никовая память. Изменить содержимое такого ПЗУ можно даже без программатора, запустив специальную программу.

Как правило, компьютер содержит микросхему ПЗУ, в которой записано встроенное программное обеспечение — набор программ, обеспечивающих проверку аппаратуры, начальную загрузку компьютера и обмен данными с некоторыми устройствами (клавиатурой, монитором, дисками). В компьютерах семейства IBM PC такое программное обеспечение называется BIOS (*англ.* Basic Input/Output System — базовая система ввода/вывода).

В IBM-совместимых компьютерах есть ещё один особый вид памяти — **память конфигурации (CMOS-память)**. В ней хранятся разнообразные настройки аппаратного обеспечения, а также часы и календарь, благодаря которым компьютер всегда «знает» текущую дату и время. Данные сохраняются благодаря питанию от небольшой батарейки. CMOS-память — это особая память, которая не входит в адресное пространство внутренней памяти. Поэтому к ней невозможно обратиться просто по адресу, и в этом смысле она скорее похожа на внешнюю память. Для работы с памятью конфигурации в ПЗУ современного ПК предусмотрена специальная программа (она называется **BIOS Setup**), причём работать с ней пользователь может только до загрузки операционной системы (при включении компьютера).

Внешняя память

Внешняя память — часть памяти компьютера, которая используется для *долговременного хранения* программ и данных.

Этот вид памяти позволяет повторно использовать программы и данные. Благодаря этому текст достаточно набрать один раз, а цифровые фотографии можно рассматривать в течение многих лет.

Устройства внешней памяти часто называют *накопителями*. К ним относятся, например, накопители на магнитных и оптических дисках, а также современные внешние запоминающие устройства на основе полупроводниковой флэш-памяти.

Внешняя память любого типа состоит из некоторого **носителя** информации (например, диска или полупроводникового кристалла) и электронной схемы управления (**контроллера**).

Компьютерный носитель информации — это средство длительного хранения данных в компьютерном формате. Носитель может быть съёмным (как в накопителях на оптических дисках), а может быть помещён внутрь неразборного устройства (жёсткий магнитный диск — «винчестер»).

Магнитные и оптические диски для обеспечения доступа к любому блоку данных быстро вращаются, а читающая головка перемещается вдоль радиуса диска. В более современных видах внешней памяти, где носителем информации является полупроводниковый кристалл, никаких движущихся частей нет, а для чтения и записи данных используются только электрические импульсы (аналогично ОЗУ).

В переносных устройствах внешней памяти, например во внешних жёстких дисках и флэш-накопителях, носитель и схема управления объединены в единый блок. Такие устройства подключаются к компьютеру снаружи через разъём.

Центральный процессор не может непосредственно обращаться к данным на носителе, он работает с ними через контроллер внешней памяти. На рисунке 5.16 схематично показано, как читаются данные с внешнего носителя информации и записываются в ОЗУ[1].

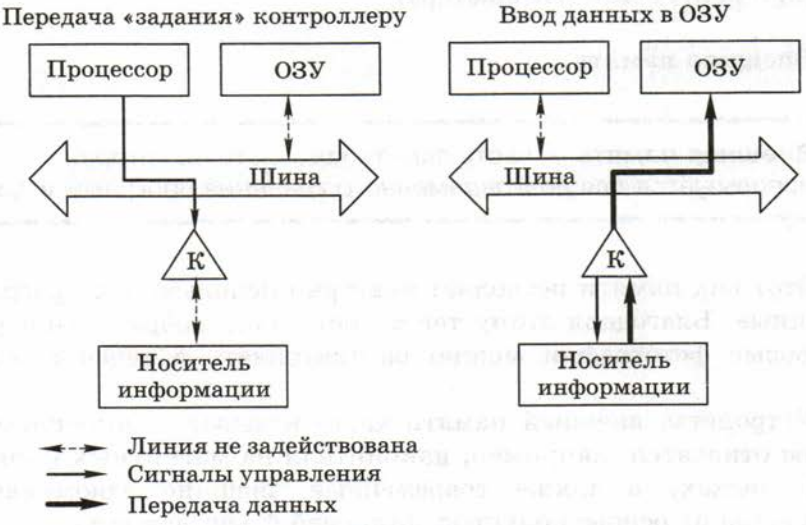


Рис. 5.16

[1] В действительности процесс обмена более сложен, в нём участвует ещё и контроллер ПДП.

Для связи с контроллером процессор использует **порты** — регистры контроллера, к которым процессор может обратиться по номеру. Процессор передаёт контроллеру «задание» на передачу данных, и контроллер берёт руководство процессом на себя. В это время центральный процессор может параллельно выполнять программу дальше или решать другую задачу. Таким образом, выполнить чтение (и запись) данных из внешней памяти гораздо сложнее, чем из внутренней памяти.

Для внешней памяти характерны следующие черты:

- обменом данными управляют контроллеры;
- прежде чем процессор сможет непосредственно использовать программу или данные, хранящиеся во внешней памяти, их нужно предварительно загрузить в ОЗУ;
- данные располагаются блоками (на дисках их принято называть *секторами*); блок данных читается и пишется как единое целое, что существенно ускоряет процедуру обмена; работать с частью блока невозможно.

В качестве внешней памяти используются самые разные носители. Первоначально программы и данные сохранялись на **бумажных перфокартах** (рис. 5.17) и **перфолентах**. Подписанные обычной ручкой или карандашом, они сортировались программистами вручную. Затем произошёл переход к **магнитным носителям**: магнитным лентам, барабанам и дискам.

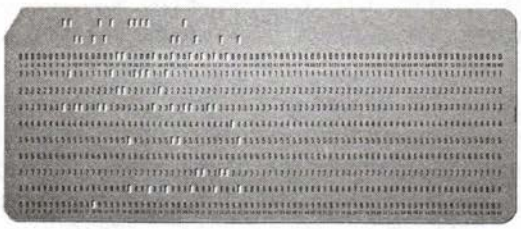


Рис. 5.17. Перфокарта

На **магнитных дисках** биты данных хранятся в виде небольших намагниченных (или ненамагниченных) областей. **Секторы** размещаются на концентрических окружностях (имеющих общий центр), которые называются **дорожками**. Поскольку длина дорожки зависит от положения на диске, количество секторов на до-

рожках может быть разным. Доступ к секторам диска — произвольный, максимальная скорость достигается тогда, когда читаемые или записываемые секторы располагаются подряд.

Управление такой сложной системой очень трудоёмко — поэтому, как нам уже известно из истории вычислительной техники, появление магнитных дисков вызвало создание специального ПО для работы с ними — операционных систем (ОС). ОС берёт на себя все технические детали, предоставляя пользователю работу с некоторыми наборами данных — **файлами**. Таким образом, начиная с дисковых накопителей, наличие **файловой системы** — это характерная черта внешней памяти, которая существенно отличает её от внутренней.

Следующей технологией хранения данных стали **оптические компакт-диски**, **CD** (англ. *Compact Disk*). При записи данных (одним из способов) луч лазера «выжигает» на поверхности диска дорожку, в которой чередуются впадины и возвышения. При считывании также применяется луч лазера, только меньшей интенсивности, чтобы не разрушить данные. Для распознавания нулей и единиц используется различное отражение от перепадов глубины и ровной поверхности диска. В отличие от магнитных дисков, где информация хранится на отдельных замкнутых дорожках, данные на оптическом диске записываются вдоль непрерывной спирали, как на старых грампластинках[1].

Сейчас широко используются оптические диски следующих поколений: **DVD** (англ. *Digital Versatile Disk* — цифровой многоцелевой диск, ёмкость до 17 Гбайт) и **Blu-ray-диски** (ёмкостью до 500 Гбайт). Они имеют тот же диаметр, что и CD-диски, но для повышения плотности записи используют лазер с меньшей длиной волны.

Были разработаны также комбинированные **магнитооптические диски**. Носителем информации в них служит магнитное вещество. При нагреве лазером оно плавится, частицы среды ориентируются в магнитном поле, и меняются оптические свойства поверхности диска. После восстановления нормальной температуры такие диски необычайно устойчивы к внешним воздействиям. Тем не менее они не получили распространения из-за высокой стоимости и малой скорости записи.

---

[1] В отличие от грампластинок спираль раскручивается от центра к краям.

Отметим, что на всех видах дисков есть разметка на секторы, благодаря которой контроллер может быстро находить нужную информацию. Сами данные помещаются между «заголовком» сектора и его завершающей записью.

Наконец, последнее достижение в области устройств внешней памяти — **запоминающие устройства на базе флэш-памяти**. В них нет движущихся частей, а носителем информации служит полупроводниковый кристалл. Данные во флэш-памяти обновляются только блоками, но для устройств внешней памяти это вполне естественно. Максимальное количество перезаписей данных для каждого блока хотя и велико, но всё же ограничено. Поэтому встроенный контроллер при записи использует специальный алгоритм для выбора свободных блоков, стараясь загружать секторы диска как можно более равномерно.

Кроме широко распространённых флэш-дисков (сленговое название — «флэшки») этот вид памяти используется в картах памяти (рис. 5.18) для фотоаппаратов, плееров и мобильных телефонов, а также в твёрдотельных винчестерах SSD (англ. *Solid State Disk*). Напомним, что ПЗУ также может изготовляться на базе флэш-памяти.



Рис. 5.18. Флэш-карта

Взаимодействие разных видов памяти

Итак, мы познакомились с разными видами внутренней и внешней памяти. Осталось разобраться, как они взаимодействуют между собой.

Иерархия памяти. Кэширование. Как следует из обсуждения в § 32, невозможно создать память, которая имела бы одновременно большой объём и высокое быстродействие. Поэтому используют многоуровневую (иерархическую) систему из нескольких типов памяти. Как правило, чем бóльший объём имеет память, тем медленнее она работает.

Самая быстрая (и очень небольшая) память — это регистры процессора. Гораздо больше по объёму, но заметно медленнее внутренняя память (ОЗУ и ПЗУ). Далее следует огромная, но ещё более медленная внешняя память. Наконец, последний уровень — это данные, которые можно получить из компьютерных сетей (рис. 5.19).



Рис. 5.19

Для редактирования файла, расположенного на диске (внешняя память), программа обработки загружает его в ОЗУ (внутренняя память), а конкретные символы, с которыми в данные доли секунды работает процессор, «поднимаются» по иерархии выше — в регистры процессора.

Производительность компьютера в первую очередь зависит от «верхних» уровней памяти — процессорной памяти и ОЗУ. Быстродействие процессоров значительно выше, чем скорость работы ОЗУ, поэтому процессору приходится ждать, пока до него дойдут данные из оперативной памяти. Чтобы улучшить ситуацию, между процессором и ОЗУ добавляют ещё один слой памяти, который называют кэш-памятью, или кэшем (от англ. *cache* — тайник, прятать).

Кэш-память — это память, ускоряющая работу другого (более медленного) типа памяти, за счёт сохранения прочитанных данных на случай повторного обращения к ним.

Кэш-память — это *статическая память*, которая работает значительно быстрее динамического ОЗУ. В ней нет собственных адресов, она работает не по фон-неймановскому принципу адресности.

При чтении из ОЗУ процессор обращается к контроллеру кэш-памяти, который хранит список всех ячеек ОЗУ, копии которых находятся в кэше. Если требуемый адрес уже есть в этом списке, то запрашивать ОЗУ не нужно и контроллер передаёт процессору значение, связанное (*ассоциированное*) с этим адресом (рис. 5.20)[1]. Такой принцип организации памяти называется **ассоциативным**.

Если нужных данных нет в кэш-памяти, они читаются из ОЗУ, но одновременно попадают и в кэш — при следующем обращении их уже не нужно читать из ОЗУ.

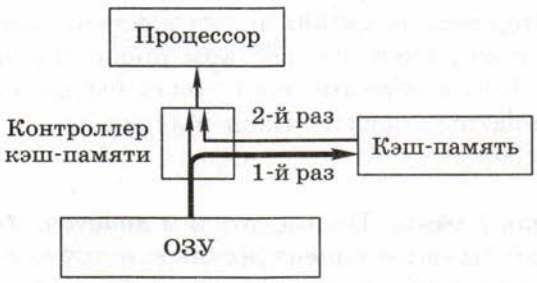


Рис. 5.20

Обычно в кэш-память заносится содержимое не только запрошенной ячейки, но и ближайших к ней (эта стрелка на рис. 5.20 показана более толстой). Таким образом, в кэше хранятся копии часто используемых ячеек ОЗУ, и передача этих данных в процессор происходит быстрее.

В работе кэш-памяти есть две основные трудности. Во-первых, объём кэша намного меньше объёма ОЗУ, и он быстро заполняется — приходится заменять наиболее «ненужные» (например, редко используемые) данные. Во-вторых, если считанные из кэш-памяти данные обрабатываются процессором и сохраняются в ОЗУ, нужно обновлять и содержимое кэша. Обе эти задачи решает контроллер кэш-памяти. Несмотря на трудности, кэширование во многих случаях повышает скорость выполнения программы в несколько раз.

Сама кэш-память также строится по многоуровневой схеме: в современных процессорах есть, по крайней мере, 2–3 уровня.

---

[1] Это напоминает поиск в Интернете содержимого документа по его названию.

Некоторые из них входят в состав процессора, а остальные выполнены в виде отдельных микросхем (поэтому на схеме многоуровневой памяти на рис. 5.18 кэш только частично расположен внутри процессора). Кэш для программ и для данных изготовляется раздельно. Это удобно потому, что считываемую программу, в отличие от данных, не принято изменять, поэтому кэш команд можно делать проще.

Подчеркнём, что **термин «кэширование»** в вычислительной технике имеет довольно широкий смысл: речь идёт о сохранении информации в более быстродействующей памяти с целью повторного использования. Например, браузер кэширует файлы, полученные из Интернета, сохраняя их на жёстком диске в специальной папке. В накопителе на жёстком диске также используется кэширование. Таким образом, кэш может быть организован как с помощью аппаратных средств (кэш процессора), так и программно (кэш браузера).

Виртуальная память. Пользователям хочется, чтобы программное обеспечение было интеллектуальным и дружественным и чтобы в нём были предусмотрены все самые мелкие детали, которые им могут потребоваться. Программистам хочется написать программу с наименьшими затратами сил и времени, поэтому они широко используют среды быстрой разработки программ (англ. **RAD** — *Rapid Application Development*). В результате программы всё больше увеличиваются в размере. Кроме того, объём обрабатываемых данных постоянно растёт. Поэтому компьютерам требуется все больше и больше памяти, особенно в многозадачном режиме, когда одновременно запускаются сразу несколько программ.

Как же согласовать эти требования с ограниченным объёмом ОЗУ? Современные операционные системы используют для этого идею **виртуальной памяти.** Предполагается, что компьютер обладает максимально допустимым объёмом памяти, с которым может работать процессор, а реально установленное ОЗУ — лишь некоторая *часть* этого пространства. Оставшаяся часть размещается в специальном системном файле или отдельном разделе жёсткого диска. Если ёмкости ОЗУ не хватает для очередной задачи, система копирует «наименее нужную» (дольше всего не использовавшуюся) часть ОЗУ на диск, освобождая необходимый объём памяти. Когда, наоборот, потребуются данные с диска, они будут возвращены в освобожденное таким же образом место ОЗУ (и это совсем не обязательно будет то самое первоначальное место!).

При использовании виртуальной памяти выполнение программ замедляется, но зато они могут выполняться на компьютере с недостаточным объёмом ОЗУ. В этом случае установка дополнительного ОЗУ может повысить быстродействие во много раз.

Использование виртуальной памяти ещё раз подтверждает, что деление памяти на внутреннюю и внешнюю память — это искусственная мера. Она вызвана тем, что невозможно создать идеальную память, удовлетворяющую всем требованиям сразу.

Основные характеристики памяти

Для пользователя важны, прежде всего, объём памяти, её быстродействие и стоимость.

Информационная ёмкость — это максимально возможный объём данных, который может сохранить данное устройство памяти.

Ёмкость памяти измеряется в тех же самых единицах, что и объём информации, т. е. в битах, байтах и производных единицах (чаще всего — в мегабайтах или гигабайтах).

Для дисков часто говорят о *форматированной* и *неформатированной ёмкости*. Первая величина — это объём «полезной» памяти, а вторая включает ещё и ту область диска, которую занимает служебная разметка.

Для оценки быстродействия памяти используют несколько величин. Любая операция обмена данными включает не только саму передачу данных, но и подготовительную часть. Это может быть, например, поиск нужного сектора диска или установка адреса внутри микросхемы ОЗУ. Время подготовки соизмеримо со временем передачи, так что пренебрегать им нельзя. Общее время обмена данными от начала подготовки до окончания передачи называют временем доступа.

Время доступа — интервал времени от момента посылки запроса информации до момента получения результата на шине данных.

При измерении этой величины обычно рассматривают самый сложный случай, когда данные считываются или записываются

в случайных местах памяти. На практике байты или секторы часто читаются по порядку, поэтому время ввода или вывода уменьшается.

Для ОЗУ время доступа измеряется в наносекундах (1 нс = 10^{-9} с), а для жёстких дисков — в миллисекундах (1 мс = 10^{-3} с). Такая разница связана с тем, что дисковод должен сначала переместить считывающую головку в нужное положение.

Поскольку устройства внешней памяти работают с целыми блоками данных, для их характеристики требуется дополнительный показатель.

Средняя скорость передачи данных — это количество передаваемых за единицу времени данных после непосредственного начала операции чтения (т. е. без учёта подготовительной стадии).

Эта характеристика обычно измеряется в мегабайтах в секунду (Мбайт/с).

Для дисковых накопителей часто указывают **частоту вращения** (в оборотах в минуту). Чем быстрее вращается диск, тем выше может быть скорость считывания и записи.

Для оценки **стоимости памяти** используют отношение стоимости модуля памяти к его информационной ёмкости. Часто говорят о стоимости одного бита или *стоимости одного гигабайта*.

Вопросы и задания

1. Зачем в компьютере нужна память?
2. С какой целью память делится на внутреннюю и внешнюю?
3. Верно ли, что внешняя память располагается вне корпуса компьютера? Приведите примеры.
4. Назовите все виды компьютерной памяти, которые вы знаете. Зачем они используются? Какими свойствами обладают?
5. К каким видам памяти применим принцип адресности фон Неймана?
6. Что означает термин «произвольный доступ к памяти»?
7. Зачем нужно ПЗУ в компьютере? Можно ли при необходимости изменить его содержимое на домашнем компьютере?
8. Что такое носитель информации? Какие носители вы можете назвать?
9. Какими носителями внешней памяти вы пользовались? Каков их объём и какую примерно его часть вы использовали?

10. Перечислите все известные вам виды дисков.
11. Что такое сектор диска?
12. Можно ли считать с диска отдельно взятый байт? Как его всё-таки получить?
13. Какую роль играет контроллер при считывании данных с диска?
14. Почему любую программу перед выполнением требуется загрузить в оперативную память?
15. Что такое флэш-память и в каких запоминающих устройствах она используется?
16. Какие разновидности полупроводникового ОЗУ существуют? Что служит в них запоминающим элементом? Каковы свойства названных вами типов ОЗУ?
17. Для чего создана кэш-память, как она работает и как повышает производительность компьютера?
18. Может ли программа обращаться к ячейкам кэш-памяти? Подумайте, относится ли кэш-память к архитектуре компьютера. Почему?
*19. Почему кэш называют ассоциативной памятью? Сравните с человеческой памятью, которую тоже часто называют ассоциативной.
20. Кэширование — это аппаратный или программный приём? Приведите примеры.
21. Перечислите все известные вам уровни иерархии компьютерной памяти и кратко охарактеризуйте их. Как меняются объём и быстродействие памяти при переходе на другой уровень иерархии (вверх или вниз)?
22. Почему компьютерам требуются все большие объёмы памяти?
23. Как работает механизм виртуальной памяти?
*24. Чем ограничен объём виртуальной памяти?
25. Какие основные характеристики используются для памяти? В каких единицах они измеряются?
26. Какая характеристика используется только для внешней памяти?

Подготовьте сообщение

а) «Устройства памяти разных поколений компьютеров»
б) «Кэш-память в современных процессорах»
в) «Виртуальная память»
г) «Что такое BIOS?»

Задачи

1. Определите информационный объём каждого вида памяти в вашем домашнем компьютере (ОЗУ, кэш-память, жёсткий диск, примерный суммарный объём CD- и DVD-дисков с данными и т. п.). Оцените отношение объёмов этих уровней памяти.

2. Сравните приведённые в тексте данные о времени доступа для ОЗУ и дисковых накопителей: на сколько порядков они различаются?
3. Пусть ёмкость жёсткого диска составляет 137 438 953 472 байта. Пользуясь калькулятором, переведите это значение в гигабайты по правилам, принятым в информатике. Сравните ответ с тем числом, которое получится, если коэффициенты при переводе принять равными 1000 (так делают, например, в физике: в 1 кг ровно 1000 г!). Какие конфликты в связи с этим могут возникнуть (и реально возникали!) у потребителей и фирм — изготовителей жёстких дисков?

§ 36 Устройства ввода

Что относится к устройствам ввода?

Информация в компьютер может вводиться с помощью самых разнообразных устройств, но не каждое из них называют устройством ввода. Например, *не являются* устройствами ввода устройства внешней памяти, рассмотренные в предыдущем разделе. Приём данных по сети также не является вводом, поскольку здесь (как и в случае внешней памяти) данные уже были введены ранее и сохранены в компьютерном формате, а теперь просто копируются.

Но, даже исключив из рассмотрения все перечисленные устройства, мы по-прежнему будем иметь довольно пёструю картину. Сравните между собой подключённый к компьютеру датчик температуры, веб-камеру, осуществляющую съемку для круглосуточной трансляции городского пейзажа в Интернет, мышь, которой пользователь запускает программы или выбирает из меню требуемое действие, и, наконец, клавиатуру, с помощью которой можно не только набирать текст, но и отдавать команды компьютеру.

Устройством ввода называется устройство, которое:
- позволяет человеку отдавать компьютеру команды и/или
- выполняет первичное преобразование данных в форму, пригодную для хранения и обработки в компьютере.

К устройствам ввода относятся:

- клавиатура;
- манипуляторы: мышь, трекбол, сенсорная панель, джойстик, трекпойнт;
- сканер;
- микрофон, видеокамера и другие источники мультимедийных данных;
- световое перо и графический планшет — специализированные устройства ввода графической информации;
- датчики.

Заметим, что некоторые устройства ввода, например датчики и веб-камеры, работают без непосредственного участия человека.

Клавиатура

Одним из первых устройств ввода была **клавиатура**. С её помощью человек вводит в компьютер текст. Текст может быть записью числа: тогда компьютер по программе преобразует текстовую строку в соответствующее двоичное число, с которым может работать процессор.

Кроме **символьных клавиш** на клавиатуре есть дополнительные **(управляющие) клавиши**. Значения некоторых из них жёстко заданы (например, клавиш управления курсором, клавиш Page Up, Home, Delete, Print Screen и др.), функции других (в первую очередь, функциональных клавиш F1–F12) программист может назначить сам. Клавиши Shift, Caps Lock, Ctrl и Alt изменяют результат нажатия остальных клавиш. С их помощью можно, например, вводить заглавные буквы.

В простейших типах клавиатур при нажатии клавиши соединяются два контакта и замыкается электрическая цепь. Роль контактов в наиболее распространённых моделях играет специальное токопроводящее напыление, наносимое на гибкую изолирующую полимерную пленку. Более качественные клавиатуры могут использовать, например, *герконы* (герметичные контакты), срабатывающие от приближающегося к ним магнита. Ещё один вариант — это ёмкостные клавиатуры, где при нажатии клавиши сближаются две небольшие пластины, образующие конденсатор. Ёмкостные клавиатуры более долговечны, так как в них нет механического контакта деталей.

Работой современной клавиатуры руководит встроенный в неё **микроконтроллер**, который:

- опрашивает все клавиши и фиксирует изменение их состояния: нажатие или отпускание;
- временно (до момента передачи в центральный процессор) хранит коды нескольких последних нажатых или отпущенных клавиш (*скан-коды*)[1];
- при наличии данных посылает требование прерывания центральному процессору и затем (по его запросу) передаёт имеющиеся данные;
- управляет световыми индикаторами клавиатуры;
- выполняет диагностику неисправностей клавиатуры.

Контроллер клавиатуры выполняет лишь минимальную обработку информации: в компьютер уходят исключительно данные о нажатии или отпускании клавиши с заданным номером. Распознавание кода набранного символа с учётом состояния клавиш сдвига выполняет программа, принимающая данные. Такое решение в очередной раз показывает, что аппаратная часть компьютера всегда делается максимально универсально, а все особенности работы компьютера определяются программным обеспечением.

Клавиатура имеет определённые технические характеристики, такие как усилие нажатия клавиш (в ньютонах) и ход клавиш (в миллиметрах).

Манипуляторы

Для ввода команд и данных в компьютер широко используются **манипуляторы** — разнообразные по конструкции устройства, воздействуя на которые (путём их перемещения, давления на их чувствительную поверхность и т. п.) пользователь может управлять компьютером, не набирая текста.

Общая идея работы манипуляторов заключается в следующем. На экране монитора отображается указатель **(курсор)**, с помощью которого человек может «показать» компьютеру интересующий его объект. Манипулятор перемещает курсор вслед за определёнными перемещениями руки человека. Когда курсор установлен в нужное место экрана, человек сообщает об этом компьютеру, обычно нажимая кнопку на манипуляторе. В принципе

[1] Скан-коды представляют собой номера клавиш и не имеют ничего общего с кодовыми таблицами символов, изученными в § 15.

манипулятор позволяет даже набирать тексты, используя нарисованную на экране клавиатуру.

Самый распространённый манипулятор — **компьютерная мышь**. Это название принято связывать с кабелем («хвостом»), соединяющим устройство с компьютером. Многим современным мышам «хвост» уже не нужен: они передают данные о своём движении с помощью электромагнитных волн (к компьютеру при этом подсоединяется специальное устройство для приёма и декодирования радиоволн — **адаптер**). Такие мыши более удобны, хотя стоят дороже и используют дополнительные элементы питания (батарейки или аккумуляторы).

Первоначально датчики движения мыши были механическими: при перемещении мыши вращался находящийся внутри неё шарик. Шарик, в свою очередь, вращал два взаимно перпендикулярных колёсика, и их поворот фиксировался электронным устройством. Полученная информация об изменении координат передавалась в компьютер. Такая механическая конструкция была неудобна, так как шарик и колесики приходилось часто очищать от пыли и грязи.

Оптические мыши, которые используются сейчас, не содержат механических частей, поэтому они долговечны и обладают высокой точностью. Расположенная «под брюхом» мыши миниатюрная видеокамера снимает изображение поверхности стола через небольшие промежутки времени (для подсветки используется светодиод или портативный лазер). Сравнивая полученные картинки, специальный микропроцессор вычисляет перемещение мыши по двум осям координат. Этот метод даёт плохие результаты, когда поверхность очень гладкая и однородная (например, стекло). В таких случаях значительно лучше работают лазерные мыши, потому что подсветка лазером даёт более контрастное изображение.

Наиболее интересная характеристика оптической мыши — это *разрешение оптического сенсора* (видеокамеры). Оно определяется как количество точек, которые способно различить устройство на отрезке заданной длины. Чем выше разрешение, тем точнее мышь способна отслеживать перемещение (это важно, например, при точной обработке изображений в графическом редакторе). Разрешение измеряется в точках на дюйм (англ. **dpi** — ***d**ots **p**er **i**nch*). Обычное разрешение мыши — около 1000 dpi, а у некоторых особо «точных» экземпляров — в несколько раз больше.

Кроме разрешения на качество работы мыши влияет *количество кадров*, которые делает видеокамера за одну секунду (до десяти тысяч). Размеры каждого кадра определяются датчиком, обычно они находятся в пределах от 16 × 16 до 30 × 30 пикселей. Зная эти данные, можно найти скорость обработки изображения в мегапикселях в секунду (Мп/с). Для игровых мышей важна также *максимальная скорость движения* — она может достигать нескольких метров в секунду.



Рис. 5.21. Трекбол (www.mousearena.com)

Шаровой манипулятор — **трекбол** (рис. 5.21) — это перевёрнутая мышь. Его чувствительный элемент — закреплённый шар, который вращается вокруг своего центра. Название «трекбол» происходит от английских слов track — направляющее устройство и ball — шар. Для портативных компьютеров он удобнее мыши, потому что не требует дополнительного ровного пространства. Кроме того, трекболы могут работать там, где есть вибрация. Сейчас трекболы практически не используются.

В ноутбуках в качестве встроенного «заменителя» мыши устанавливают еще один тип манипулятора — **сенсорную панель** (*англ.* touchpad), воспринимающую движение по ней пальца. Панель состоит из небольшой чувствительной к давлению поверхности и двух кнопок. Короткое касание чувствительной панели заменяет щелчок мышью (можно использовать также кнопки рядом с панелью). Современные панели способны воспринимать не только перемещение, но и другие команды. Например, для прокрутки документа можно проводить пальцем вдоль правой или нижней границы панели (там, где в окне принято располагать полосы прокрутки). Некоторые панели даже способны анализировать касание в нескольких точках (режим *мультитач*, от англ. *multi-touch* — множественное касание).

«Менее серьёзный» манипулятор — **джойстик** (англ. *joy stick* — «весёлая» рукоятка) — используется в основном в компьютерных играх и может быть оформлен самым причудливым образом. Джойстик имеет ручку, при повороте которой внутри корпуса замыкаются контакты, соответствующие направлению наклона ручки. В некоторых моделях дополнительно установлен датчик давления, и чем сильнее пользователь наклоняет ручку, тем быстрее движется указатель по экрану.

В некоторых ноутбуках в середине клавиатуры устанавливается **трекпойнт** (это слово можно перевести с английского как указатель курса или маршрута). Трекпойнт (рис. 5.22) — это кнопка, которая определяет направление давления пальца и преобразует эту команду в перемещение курсора на экране.



Рис. 5.22. Трекпойнт (www.globalnerdy.com)

Сканер

Сканер — это устройство для ввода в компьютер графической информации.

С его помощью можно преобразовать в компьютерные данные рисунки, фотографии, снимки на фотоплёнке (негативы и слайды), а также получить снимки объектов не слишком большой толщины.

Часто с помощью сканера в компьютер вводят офисные документы. При этом сканер передаёт в компьютер изображение документа в виде картинки. Чтобы отсканированный текст можно было редактировать, нужно превратить эту картинку в коды символов. Для этого используют **программы оптического распознавания символов** (англ. **OCR** — *Optical Character Recognition*). OCR-программы пытаются «угадать» в пикселях рисунка очертания букв, и определить, какие это буквы, сверяя контуры с имеющимися у них образцами. В принципе, можно распознавать и рукописный текст, однако программы справляются с ним значительно хуже, чем с печатным (подумайте почему?).

Принцип работы сканера достаточно прост. Луч света от яркого источника пробегает вдоль сканируемой поверхности, а светочувствительные датчики при этом воспринимают отражённые лучи и определяют их интенсивность и цвет. Можно сказать, что сканер — это очень сильно упрощённый цифровой фотоаппарат.

Сканеры могут иметь разную конструкцию:

- *ручные*, где считывающую головку перемещает пользователь (вспомните, как считываются штрих-коды в магазине);
- *планшетные*, в которых неподвижный объект кладётся на стекло, а светочувствительная головка перемещается внутри сканера;

- *рулонные*, протягивающие бумагу с изображением мимо неподвижной головки;
- *барабанные*, где сканируемый объект наклеивается на вращающийся барабан, который медленно вращается мимо неподвижной головки; такие сканеры обеспечивают наилучшее качество сканирования и применяются в издательской деятельности.

Сканеры часто объединяют в одном корпусе с лазерным принтером, копировальным аппаратом и факсом — получается **многофункциональное устройство (МФУ)**.

Самая важная характеристика сканера — разрешающая способность.

Разрешающая способность — это максимальное количество точек на единицу длины, которые способен различить сканер.

Разрешающая способность сканера измеряется в пикселях на дюйм (англ. **ppi** — *pixels per inch*). При сканировании совсем не обязательно устанавливать максимально возможное разрешение. Конечно, чем оно выше, тем лучше качество, но зато и файл займёт больше места на диске! Рекомендуемое разрешение зависит от того, зачем сканируется материал (табл. 5.2).

Таблица 5.2

Применение	Разрешение, ppi
Сканирование в отражённом свете:	
иллюстрации для веб-страниц	75–150
сканирование текста без распознавания	150–200
сканирование текста для распознавания	300–400
цветное фото для печати на струйном принтере	200
цветное фото для типографской печати	не менее 300
Сканирование в проходящем свете:	
35-мм плёнка, для веб-страниц	200–600
35-мм плёнка, для печати на струйном принтере	600–2000

Другая важная характеристика режима сканирования — **глубина цвета (разрядность)**, т. е. количество двоичных разрядов, которое используется для кодирования цвета одного пикселя (см. § 16). Если для кодирования цвета использовано N двоичных разрядов, то общее количество возможных цветов равно 2^N. Высококачественные устройства позволяют сканировать изображения с глубиной цвета 48 битов.

Цифровые датчики

Датчик — устройство, регистрирующее какую-либо физическую величину и преобразующее её в сигналы (обычно электрические).

Компьютер способен не просто хранить большое количество данных, полученных от многочисленных датчиков, но и проводить их математическую обработку. Таким образом, на основе компьютера может быть построена мощная цифровая лаборатория.

Многие датчики вырабатывают аналоговые данные. Поэтому для их подсоединения к компьютеру необходимо устройство, преобразующее аналоговые сигналы в цифровые — **аналого-цифровой преобразователь (АЦП)**.

Вопросы и задания

1. Зачем нужны устройства ввода?
2. Можно ли сетевую карту, через которую компьютер получает данные, назвать устройством ввода? Почему?
3. Перечислите все известные вам устройства ввода. С какими из них вы работали?
4. Какие данные можно ввести в компьютер с помощью клавиатуры?
5. Что такое управляющие клавиши? Зачем они используются?
6. Что такое функциональные клавиши?
7. Нажатие одной и той же клавиши вызывает разную реакцию компьютера в зависимости от состояния клавиш сдвига — Shift, Alt и Ctrl. Сколько различных команд можно ввести с помощью одной основной клавиши, используя клавиши сдвига?
8. Работая в электронной таблице, пользователь нажал клавиши «3», «2» и «1». Какие операции должен выполнить компьютер, чтобы соответствующее число было записано в память?
9. Зачем в клавиатуре установлен микроконтроллер?
10. Почему клавиатура не передаёт в компьютер готовые коды символов?
11. Как выполняется локализация (использование национальных символов) на клавиатуре? Найдите сведения по этому вопросу в Интернете.

12. Что такое манипулятор?
13. Какие разновидности манипуляторов вы знаете? С какими из них вы непосредственно работали? Сравните приёмы работы с ними. Обоснуйте свой ответ.
14. Как устроена компьютерная мышь?
*15. В старых моделях мышей вращение колёсиков фиксировалось с помощью источника света и фотодатчика: при прохождении прорези на диске датчик фиксировал появление света. Докажите, что с одним таким датчиком можно зафиксировать движение и даже измерить его скорость, но нельзя определить направление вращения. Предложите вариант, позволяющий решить эту проблему.
16. Каким образом движение мыши управляет перемещением курсора на экране?
17. Вспомните все известные вам приёмы работы с мышью.
18. Что такое беспроводная мышь?
19. Что такое трекбол и как он работает?
20. Что такое сенсорная панель?
21. Как устроен джойстик?
22. Что такое трекпойнт?
23. Что можно делать с помощью сканера?
24. Какие бывают сканеры по конструкции?
25. Можно ли с помощью сканера получить фотографию реального объекта?
26. Как происходит распознавание отсканированного текста?
27. Назовите наиболее важные характеристики сканеров.
28. Какое устанавливать разрешение при сканировании? На что оно повлияет?
29. Что такое датчики? Каковы возможности компьютера в автоматизации эксперимента?

Подготовьте сообщение

а) «Принципы работы сканеров»
б) «Беспроводные устройства ввода»
в) «Сенсорные устройства ввода»
г) «Цифровые лаборатории»

Задачи

1. Видеокамера оптической мыши имеет размер кадра 16 × 16 пикселей, за одну секунду обрабатывается 9000 кадров. Рассчитайте скорость обработки данных в мегапикселях в секунду.
2. Вычислите, сколько точек получится при сканировании изображения размером 10 × 10 см с разрешением 300 ppi. Оцените объём полученного файла при сканировании в режиме 256 оттенков серого. Проделайте аналогичную оценку для режима 24-битного цвета.

§ 37 Устройства вывода

Устройства вывода — это устройства, которые представляют компьютерные данные в форме, доступной для восприятия человеком.

Первыми устройствами вывода были **панели индикаторных лампочек**. Каждая из них показывала состояние отдельного бита: горящая лампочка обозначала единицу, а выключенная — ноль. Для чтения результата нужно было хорошо знать двоичную систему.

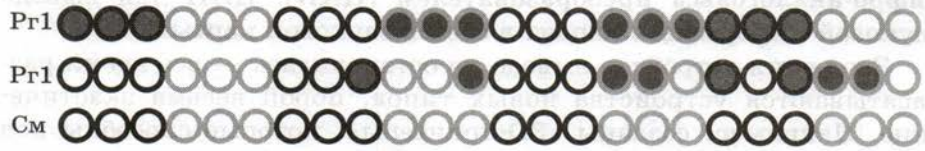


Рис. 5.23

Схематический рисунок 5.23 изображает индикаторную панель на пульте ЭВМ первых поколений. Разноцветные колпачки патронов с лампочками помогали правильно считывать результат: каждая группа из трёх битов — это одна восьмеричная цифра. Если считать, что горящие лампочки на рисунке обозначены тёмным цветом, то в регистре Рг1 читается восьмеричное число 70070770_8, а сумматор См очищен (заполнен нулями).

Такие панели использовались для обслуживающего персонала вплоть до третьего поколения ЭВМ, однако для большинства пользователей такой вывод данных был непонятен. Первые «настоящие» устройства вывода печатали числа в десятичном виде на бумагу. Затем **печатающие устройства** научились печатать не только цифры, но и буквы. Они работали по принципу печатающей машинки: рельефный шаблон символа ударял по красящей ленте, прижатой к бумаге, и оставлял отпечаток.

Кроме устройств, печатающих символы, появились **графопостроители (плоттеры)**, которые рисовали перьями на бумаге графики функций и простейшие картинки. Так как современные

принтеры могут выводить текст и графику (в том числе и цветную), специальные устройства для вывода графики практически не используются.

Революционным событием стало создание **мониторов**. Это позволило избавиться от ненужного расхода бумаги — теперь можно было выводить на печать только самое необходимое. Кроме того, управление и обслуживание ЭВМ стало более удобным.

Компьютеры четвёртого поколения начали обрабатывать мультимедийную информацию — звуковые и видеоданные. Поэтому к компьютерам стали подключать устройства для вывода такой информации: **звуковые колонки**, **наушники**, **телевизор** и т. п. Некоторые из этих устройств (например, наушники) — аналоговые, поэтому для вывода необходимо преобразовать дискретные компьютерные данные в аналоговую форму. Для этого используется специальная электронная схема, которую называют **цифро-аналоговым преобразователем (ЦАП)**. ЦАП для вывода звуковой информации входит в состав звуковой карты.

Эволюция устройств вывода не остановилась — всё время разрабатываются устройства новых типов, порой весьма экзотические. Например, созданы «3D-принтеры», которые способны под управлением компьютера создавать объёмные тела из различных материалов (прежде всего из пластика).

Монитор

Компьютерный **монитор** состоит из **дисплея** (панели, на которую смотрит человек) и **электронных схем**, позволяющих выводить на этот дисплей текстовую и графическую информацию.

Мониторы во многом используют телевизионные технологии. В конце XX века для компьютеров применялись мониторы на основе **электронно-лучевых трубок (ЭЛТ)**, но они были вытеснены **жидкокристаллическими (ЖК) мониторами**, которые обладают рядом *преимуществ*:

- малый вес и размеры;
- в 2–4 раза меньшее потребление электроэнергии;
- нет искажений изображения, характерных для электронно-лучевых трубок;
- значительно ниже уровень электромагнитного излучения.

Тем не менее ЖК-мониторы имеют *недостатки*, которые трудно устранить в производстве:

- цветопередача хуже, чем у ЭЛТ-мониторов; например, очень трудно получить чисто чёрный цвет;
- контраст и цвета изображения меняются в зависимости от угла, под которым мы смотрим на монитор;
- при быстром изменении изображения заметно «запаздывание» (жидкие кристаллы не могут поворачиваться слишком быстро);
- при существующих технологиях изготовления у многих ЖК-мониторов есть дефектные точки, которые не работают (так называемые «битые пиксели»);
- ЖК-мониторы могут отображать чёткую картинку только в одном разрешении, совпадающем с размерами матрицы.

Независимо от конструкции, экран любого монитора строится из отдельных *точек*, причём каждая из них образована близко расположенными областями трёх основных цветов — красного, зелёного и синего. Для ЖК-монитора эти области имеют форму прямоугольников, слегка вытянутых по вертикали. Расстояние между их центрами — доли миллиметра, поэтому глаз человека воспринимает все три составляющие как одну точку «суммарного» цвета.

Элементы экрана часто называют пикселями, что не совсем точно. Строго говоря, пиксель — это точка рисунка, а не экрана. Например, на одном и том же мониторе можно легко устанавливать разные размеры рабочего стола в пикселях. Компьютер «проектирует» пиксели картинки на экран с учётом установленного разрешения. При этом одному пикселю может соответствовать одна или несколько точек экрана.

Элемент ЖК-экрана — это жидкий кристалл, способный под воздействием электрического напряжения менять свои оптические свойства. Каждая точка управляется своим полупроводниковым транзистором. Подчеркнём, что сам жидкий кристалл не способен светиться, он лишь регулирует пропускание света от расположенного за ним источника света.

Наиболее важные характеристики мониторов — это **размер диагонали** (в дюймах) и **максимальное разрешение** (количество точек экрана по ширине и высоте). Для ЖК-мониторов максимальное разрешение определяется количеством элементов матрицы. Если установить другое (более низкое) разрешение, то качес-

тво изображения будет хуже, так как видеосистеме придётся «растягивать» картинку на реально существующие точки.

Процессор передаёт данные для вывода **видеокарте (видеоконтроллеру)**, которая управляет выводом изображения на монитор. Современные видеокарты содержат микропроцессор для обработки графической информации **(графический ускоритель)** и собственную **видеопамять**. Можно считать, что видеокарта — это специализированный компьютер, который существенно ускоряет построение и вывод на монитор графических изображений, особенно трёхмерных.

Инженеры активно работают над созданием всё более совершенных устройств вывода. Большое внимание уделяется разработке так называемых *3D-дисплеев*, которые смогут отображать информацию в трёх измерениях.

Печатающие устройства

Печатающие устройства **(принтеры)** служат для вывода текстовой и графической информации на бумагу или плёнку. Современные принтеры обрабатывают символы как графику, т. е. рисуют их. На принтерах можно печатать очень сложные изображения, в том числе цветные фотографии.

В настоящее время существуют четыре основных типа принтеров: матричные, струйные, лазерные и сублимационные.

Матричные принтеры — это последнее поколение принтеров с ударным принципом работы. Печатающая головка содержит вертикальный ряд иголок, которые под воздействием управляющих сигналов ударяют по красящей ленте, оставляя на бумаге отпечатки в виде маленьких точек (рис. 5.24).



Рис. 5.24

Головка движется в горизонтальном направлении, что позволяет сформировать строку из символов произвольного вида. На таком принтере можно получать не только тексты, но чёрно-белые рисунки, однако вывод графики происходит очень медленно. *Достоинства* матричных принтеров — дешевизна самих принтеров и расходных материалов (красящих лент), а также способность печатать практически на любой бумаге. Однако они не могут обеспечить высокое качество печати, работают медленно и сильно шумят.

Печатающая головка **струйных принтеров** содержит крошечные отверстия, через которые под большим давлением на бумагу выбрасываются чернила. Диаметр получаемых при этом точек гораздо меньше, чем у матричных принтеров, что позволяет получить значительно лучшее качество печати. В цветных принтерах чаще всего устанавливаются четыре картриджа: с голубой, пурпурной, жёлтой и чёрной красками (вспомните цветовую модель CMYK, см. § 16). Изображение строится из точек этих цветов. В некоторых моделях для повышения качества используют шесть базовых цветов. Для печати на струйных принтерах необходима качественная бумага, кроме того, напечатанное изображение расплывается при попадании воды.

Лазерные принтеры обеспечивают очень высокое качество печати. Компьютер строит в памяти полный образ страницы и передаёт его принтеру. Тот с помощью лазерного луча построчно переносит изображение на вращающийся барабан — строит электростатическую копию картинки. Затем к барабану притягиваются мелкие частицы красящего порошка — **тонера**, причём чем сильнее наэлектризован участок барабана, тем больше краски он получает. На следующем этапе бумага прижимается к барабану, в результате на ней строится отпечаток картинки. Чтобы краска не осыпалась, на выходе нагретый валик вплавляет частицы тонера в бумагу (рис. 5.25). Поскольку лазерные принтеры используют достаточно сложные технологии, они стоят дороже матричных и струйных и потребляют больше электроэнергии.

Светодиодные принтеры (их тоже часто называют лазерными) работают по такому же принципу, но изображение переносится на барабан не лазером, а светодиодной матрицей.

Сублимационный принтер печатает изображение совсем иначе: головка принтера нагревает поверхность, размягчая её, а затем «впрыскивает» крохотные частицы красителя. Сверху наносится защитный слой, который предохраняет краску от разруше-

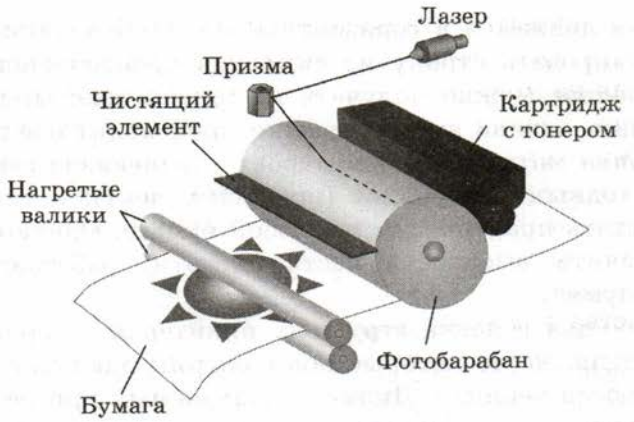


Рис. 5.25

ния солнечными лучами, и в итоге образуется очень стойкое изображение. Сублимационные принтеры прекрасно подходят для печати на пластиковых картах и компакт-дисках, часто используются для печати фотографий. Их *недостатки* — низкая скорость печати (более 1 минуты на одну фотографию) и высокая стоимость.

Важнейшей характеристикой принтера является его разрешающая способность.

Разрешающая способность принтера — это максимальное количество точек, которые он способен напечатать на единицу длины.

По традиции разрешающая способность измеряется в точках на дюйм (англ. **dpi** — *dots per inch*). Все современные струйные и лазерные принтеры имеют разрешающую способность не ниже 300 dpi, что обеспечивает высококачественную печать. Некоторые принтеры позволяют пользователю менять разрешающую способность, регулируя тем самым качество печати.

Обратим внимание на разницу обозначений ppi (пиксели на дюйм) и dpi (точки на дюйм). В ppi измеряется разрешение цифрового изображения (например, отсканированного), а в dpi — качество печати принтера. Каждый пиксель картинки может изо-

бражаться принтером в виде нескольких точек. Вспомните, что примерно то же самое происходит при выводе пикселей изображения на монитор.

Принтеры также часто сравнивают по скорости печати (в страницах в минуту). Наименьшая скорость печати у сублимационных и матричных принтеров, а наибольшая — у лазерных. Цветная печать, как правило, выполняется дольше, чем более простая чёрно-белая.

Устройства ввода/вывода

Некоторые компьютерные устройства нельзя однозначно отнести ни к устройствам ввода, ни к устройствам вывода. Пример такого «гибрида» — сенсорный экран. С одной стороны, на него выводится информация, а с другой — пользователь вводит команды, нажимая на нужный участок изображения. Сенсорные экраны применяют в портативных компьютерах, платёжных и информационных терминалах, а также для представления презентаций.

В некоторых мобильных телефонах, карманных и планшетных персональных компьютерах сенсорный экран заменил клавиатуру и занимает всю переднюю панель. Многие из этих устройств (например, смартфон *iPhone* и планшетный компьютер *iPad* фирмы Apple) используют технологию **мультитач**. Это значит, что сенсорный экран отслеживает нажатия и движения пальцев в нескольких точках одновременно. Например, сближая пальцы рук, пользователь уменьшает масштаб изображения на дисплее, а раздвигая — увеличивает.

Вопросы и задания

1. Зачем нужны устройства вывода?
2. В чём сходство и различие устройств ввода/вывода и внешней памяти?
3. Перечислите все известные вам устройства вывода. С какими из них вы работали?
4. Что такое МФУ?
5. Что такое АЦП и ЦАП?
6. Что такое монитор и каковы его возможности?
7. Что является элементом изображения в мониторе?
8. Отличается ли пиксель от точки экрана?
9. Компьютер выводит на экран монитора число, хранящееся в ячейке памяти. Какие действия он должен выполнить для этого?

10. Назовите наиболее важные характеристики мониторов.
11. Зачем нужна видеокарта? Каким образом она позволяет разгрузить центральный процессор?
12. Что такое видеопамять?
13. Какие типы принтеров вам известны? Опишите каждый из них и составьте классификационную таблицу.
14. Как работает лазерный принтер?
15. Что такое разрешающая способность принтера?
16. В чём различие единиц dpi и ppi?
17. Как пересчитать сантиметры в дюймы?
18. Что такое технология «мультитач»?

Подготовьте сообщение

а) «Жидкокристаллические мониторы»
б) «Принтеры для печати фотографий»
в) «3D-принтеры»
г) «Сенсорные экраны»
д) «Устройства для вывода звука»

Задачи

1. Рассмотрите схему индикаторной панели содержимого регистров, приведённую в начале § 37 (рис. 5.23). Прочитайте коды чисел, хранящиеся в регистрах Рг1 и Рг2. Считая числа беззнаковыми целыми, сложите их, пользуясь правилами восьмеричной арифметики. Как будет выглядеть сумматор, когда в нём появится код суммы?

Практические работы к главе 5

Работа № 12 «Моделирование работы процессора»
Работа № 13 «Процессор и устройства вывода»

ЭОР к главе 5 на сайте ФЦИОР (http//fcior.edu.ru)

- От абака до ноутбука. Поколения компьютерной техники
- Архитектура компьютера
- Архитектура машин пятого поколения
- Конфигурация компьютера. Выбор конфигурации в зависимости от решаемых задач
- Магистраль. Передача данных внутри компьютера
- Принцип открытой архитектуры

- Процессор
- Видеоплата. Звуковая плата
- Внутренняя память компьютера
- Внутренняя память компьютера. Внешняя память компьютера. Типы накопителей информации
- Устройства ввода информации
- Типы дисплеев
- Типы мониторов
- Устройства вывода информации

Самое важное в главе 5

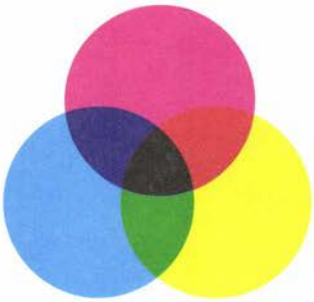
- Компьютер — это универсальный программируемый автомат для обработки данных.
- Компьютер — это универсальное устройство, которое может быть настроено для решения различных задач загрузкой соответствующего программного обеспечения.
- Основные устройства компьютера — процессор, память, устройства ввода и устройства вывода.
- Процессор — это устройство, предназначенное для автоматического считывания команд программы, их расшифровки и выполнения.
- Память — это устройство для хранения программ и данных. Различают внутреннюю память (для хранения данных решаемой задачи) и внешнюю память (для долговременного хранения данных). Невозможно создать память, которая имела бы одновременно большой объём и высокое быстродействие. В современных компьютерах используется несколько видов памяти; как правило, чем больше объём, тем ниже быстродействие.
- Устройства ввода предназначены для управления компьютером и первичного ввода данных.
- Устройства вывода предназначены для вывода данных в форме, понятной человеку.

Для заметок

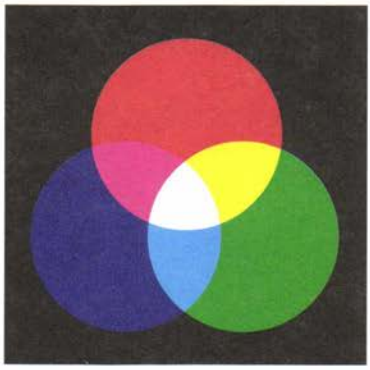
Для заметок

Для заметок

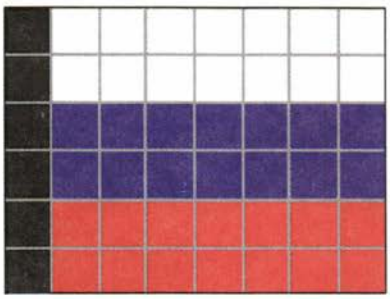
Наложение красок на бумаге в модели цвета CMYK



Модель цвета RGB



Кодирование изображения российского флага



00	11	11	11	11	11	11	11
00	11	11	11	11	11	11	11
00	10	10	10	10	10	10	10
00	10	10	10	10	10	10	10
00	01	01	01	01	01	01	01
00	01	01	01	01	01	01	01

Излучаемый и отражаемый свет

