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

$$0{,}a_1a_2a_3a_4 = a_1 \cdot p^{-1} + a_2 \cdot p^{-2} + a_3 \cdot p^{-3} + a_4 \cdot p^{-4}$$

или с помощью схемы Горнера:

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

Умножив это число на p, получаем $a_1{,}a_2a_3a_4$. Если взять целую часть результата, мы получим цифру a_1. Таким же способом можно найти оставшиеся цифры дробной части: на каждом шаге 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

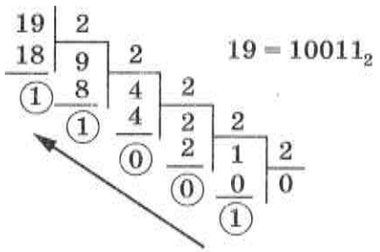
1. Запишите число $0{,}12321_4$ в развёрнутой форме и с помощью схемы Горнера.
2. Переведите число 15,125 в двоичную, четверичную, шестеричную и восьмеричную системы.
3. Какие из этих чисел больше, чем $^1/_2$: $0{,}011_2$; $0{,}12_3$; $0{,}21_4$; $0{,}22_5$; $0{,}25_6$; $0{,}35_7$; $0{,}35_8$?
4. Переведите 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, переведем в двоичную систему число 19:



$19 = 10011_2$

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, обведенных кружками. Единицы стоят в шестом, третьем, втором и нулевом разрядах:

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

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

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

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

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

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:

$$\begin{array}{r} {\scriptstyle \dot{1}\,\dot{1}0} \\ -\ 2\,1 \\ \hline 9 \\ \hline 1\,2 \end{array}$$

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 заполняются цифрой 9, это старшая цифра десятичной системы счисления:

$$\begin{array}{r} {\scriptstyle \dot{2}\,\dot{0}\,\dot{0}\,\dot{1}0} \\ -\ 2\,0\,0\,1 \\ \hline 9 \\ \hline 1\,9\,9\,2 \end{array}$$

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 берется заём) заполняются не девятками, а единицами (старшей цифрой системы счисления). Например:

$$\begin{array}{r} {\scriptstyle \dot{0}\,1\,1\,2} \\ 1\,1\,0\,0_2 \\ -\ 1_2 \\ \hline 1\,0\,1\,1\,1_2 \end{array} \qquad \begin{array}{r} {\scriptstyle \dot{0}\,1\,1\,2\,\dot{0}\,2} \\ 1\,0\,0\,0\,1\,0\,1_2 \\ -\ 1\,1\,0\,1\,1_2 \\ \hline 1\,0\,1\,0\,1\,0_2 \end{array}$$

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

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

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

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

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

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

Обеспечение точности расчётов с дробными (вещественными) числами — это очень важная и актуальная проблема, пока до конца не решённая. Поэтому сначала надо попытаться решить задачу, используя только операции с целыми числами. Например, пусть требуется 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

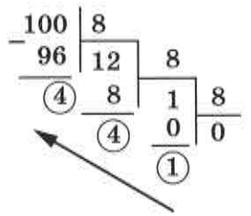
1. Переведите числа 25, 31, 37, 63, 85, 127, 128 в двоичную систему счисления.
2. Переведите числа 100011_2, 101101_2, 110111_2, 1001011_2, 1011111_2, 1101001_2 в десятичную систему счисления.
3. Сколько единиц в двоичной записи чисел 173, 195, 126, 208?
4. Сколько значащих нулей в двоичной записи чисел 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) $1011111_2 + 111011_2$;
   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 (позиционная 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 пробелами. Обратите внимание, что для всех чисел, меньших четырёх, результат перевода в двоичную систему дополняется 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), в начало числа нужно добавить два нуля (они подчёркнуты):

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

Далее по табл. 2.5 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`. Режим доступа кодируется тремя битами, которые разрешают чтение (`r`, read, старший бит), запись (`w`, write) и выполнение файла (`x`, execute, младший бит). Код $7 = 111_2$ (`rwx`) означает, что все биты установлены (полный доступ), а код $5 = 101_2$ (`r-x`) разрешает чтение и выполнение файла, но запрещает его изменение.

Вопросы и задания

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

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) $134_8 : 10111_2$;
   к) $214_8 : 1110_2$.

*6. Переведите числа 12,5 и 45,3 в восьмеричную систему счисления.

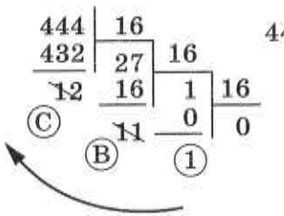
§ 13
Шестнадцатеричная система счисления

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, результат перевода в двоичную систему нужно дополнить старшими нулями до 4 знаков. Для первой цифры это делать не нужно, так как нули в начале числа не изменяют его.

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:

$$1000010000101010111100_2 =$$
$$= 10\ 0001\ 0000\ 1010\ 1011\ 1100_2 = 210ABC_{16}.$$

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:

0	1	0	1	1	1	1	0
	5				E		

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; в) 1111001101111110101_2;
   б) 1010101101010110_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) $740_8 : 18_{16}$.

*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 ($\bar{1}$);
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:

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

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

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

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

а) «Сравнение двоичной и двоично-десятичной систем счисления»
б) «Факториальная система счисления»
в) «Фибоначчиева система счисления»

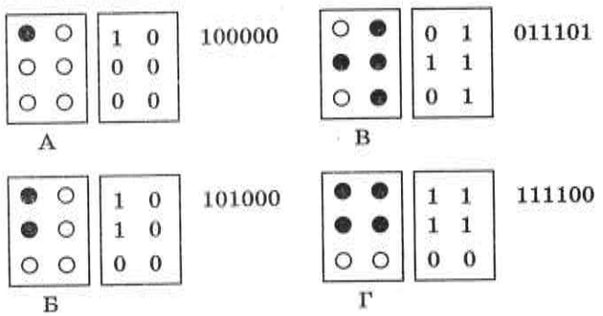
Задачи

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.

§ 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 системе Брайля любой 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 необходимое количество i двоичных разрядов так, чтобы с их помощью можно было закодировать не менее N разных последовательностей (т. е. $2^i \geq N$);

3) составляют таблицу, в которой каждому символу сопоставляют код (номер) — целое число в интервале от 0 до $2^i - 1$;
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

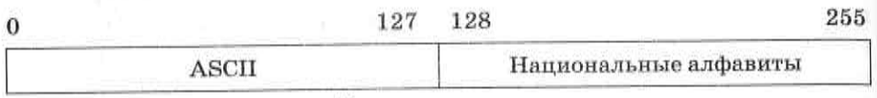
Для того чтобы упростить передачу текстовой информации, разработаны стандарты, которые закрепляют определённые коды за общеупотребительными символами. Основным международным стандартом является 7-битная **кодировка ASCII** (англ. *American Standard Code for Information Interchange* — американский стандартный 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 еще 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 Unix-совместимых операционных системах и электронной почте;
- альтернативная кодировка (CP-866) — в системе MS DOS;
- кодовая страница **MacCyrillic** — на компьютерах фирмы Apple (Макинтош и др.).

Проблема состоит в том, что если набрать русский текст в одной кодировке (например, в Windows-1251), а просматривать в другой (например, в KOI8-R), текст будет 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 русского 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 6 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.

Начнем 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).

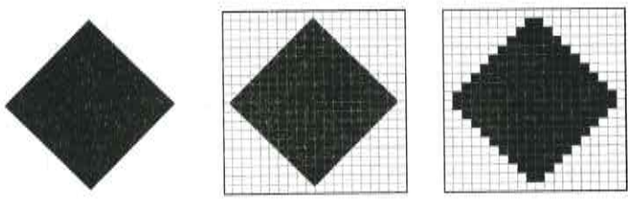


Рис. 2.15

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

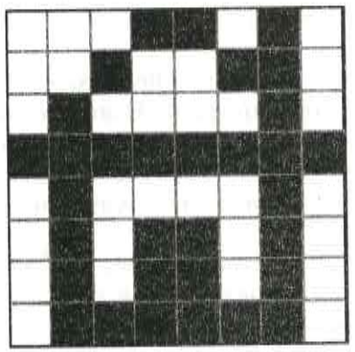
Пиксель (англ. *pixel* — picture element, элемент рисунка) — это наименьший элемент рисунка, для которого можно независимым образом задать свой цвет.

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[1];
- 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

[1] А можно сделать и наоборот, обозначив белые пиксели единицами, а чёрные — нулями.

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.

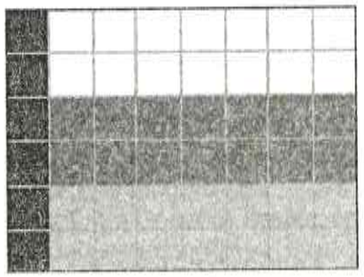
Разрешение — это количество пикселей, приходящихся на единицу линейного размера изображения.

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* (и на цветном рисунке 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. Поскольку компьютер работает только с числовыми данными, информацию о цвете для вывода на экран нужно выразить в виде числа (или набора чисел).

Обсудим подробнее, как это можно сделать, а затем вернёмся к примеру с кодированием изображения флага и завершим его.

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 определённой длины) можно имитировать, используя только три световых луча (красный, зелёный и синий) разной яркости. Следовательно, любой цвет (в том числе и «белый») приближённо раскладывается на три составляющих — красную, зелёную и синюю. Меняя силу этих составляющих, можно составить любые цвета (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).

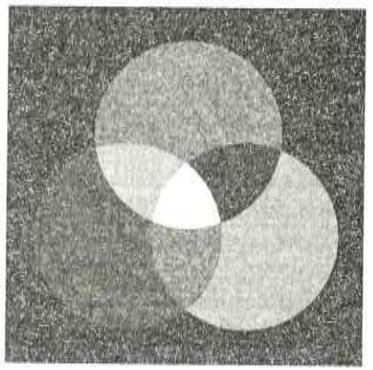


Рис. 2.18

В **модели RGB** яркость каждой составляющей (или, как говорят, каждого канала) чаще всего кодируется целым числом[1] 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 шестнадцате-

[1] Или дробным числом от 0 до 1.

ричной 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, используемых 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

[1] Пурпурный цвет получается при смешивании синего и красного.

пикселей, закодированный в режиме истинного цвета, будет занимать 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:

1) выбирается количество цветов N (как правило, не более 256);
2) из палитры истинного цвета (16 777 216 цветов) выбираются любые N цветов и для каждого из них находятся составляющие в модели RGB;
3) каждому из выбранных цветов присваивается номер (код) от 0 до $N - 1$;
4) составляется палитра: сначала записываются RGB-составляющие цвета, имеющего код 0, затем — составляющие цвета с кодом 1 и т. д.;

5) 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:

1) определить размер палитры: $3 \cdot N$ байтов, или $24 \cdot N$ битов;
2) определить глубину цвета (количество битов на пиксель), т. е. найти наименьшее натуральное число i, такое что $2^i \geq N$;
3) вычислить общее количество пикселей L, перемножив размеры рисунка;
4) определить информационный объём рисунка (без учёта палитры): $L \cdot i$ битов.

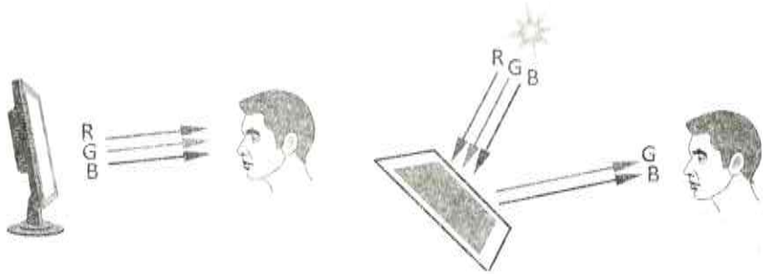
В таблице 2.9 приведены данные по некоторым вариантам кодирования с палитрой.

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 цветом 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 краску голубого цвета, красные лучи будут поглощаться, останутся только синие и зелёные. Если сверху нанести ещё

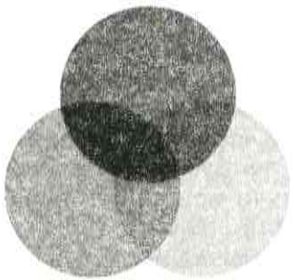


Рис. 2.20

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 чёрный, а при увеличении яркости до максимума — в белый.

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.

Форматы файлов

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 объектов;
- **GIF** (англ. *Graphics Interchange Format* — формат для обмена изображениями; файлы с расширением gif) — формат, поддерживающий только кодирование с палитрой (от 2 до 256 цветов); в отличие от предыдущих форматов, части рисунка могут быть прозрачными, т. е. на веб-странице через них будет «просвечивать» фон; в современном варианте формата GIF можно хранить анимированные изображения; используется сжатие без потерь, 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	—

Данные в файлах разных форматов кодируются по-разному. Как же компьютер выбирает нужный способ расшифровки?

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 (координаты вершин, цвет и толщина контура, тип заливки и др.) хранятся в виде чисел.

Векторный рисунок можно «разобрать» на части, растащив мышью его элементы, а потом снова собрать полное изображение (рис. 2.21).

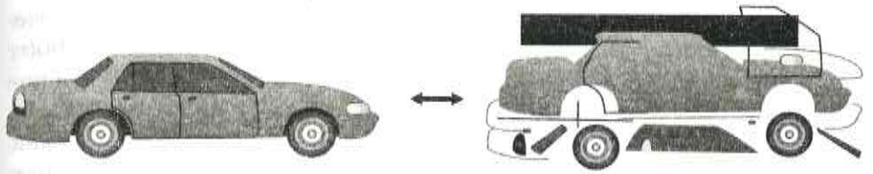


Рис. 2.21

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*.

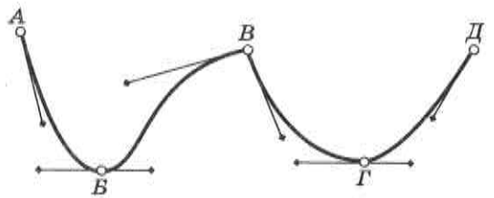


Рис. 2.22

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 рисунки обладает значительными **преимуществами** по сравнению с растровым:

- при кодировании нет потери информации, если изображение (например, чертеж, схема, карта, диаграмма) может быть полностью разложено на простейшие геометрические фигуры;
- 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

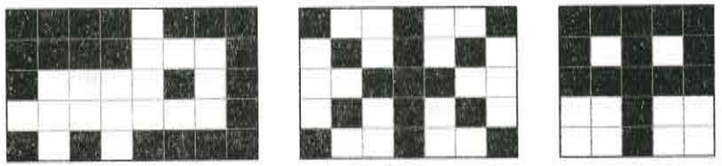
1. Какие два принципа кодирования рисунков используются в компьютерной технике?
2. Почему не удается придумать единый метод кодирования рисунков, пригодный во всех ситуациях?
3. В чём состоит идея растрового кодирования? Что такое растр?
4. Что такое пиксель?
5. Что такое дискретизация? Почему она необходима?
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?
20. Что такое кривые Безье?
21. Почему при увеличении растрового рисунка появляются «ступеньки»?
22. Что такое векторное кодирование? В чём его отличие от растрового? Каковы преимущества и недостатки растрового и векторного кодирования?
23. В каких задачах используют векторное кодирование?
24. Какие форматы векторных рисунков вы знаете? Приведите примеры.

Подготовьте сообщение:

а) «Цветовая модель Lab»
б) «Цветовая модель HSB»
в) «Цветовые профили»
г) «Преобразования между цветовыми моделями»
д) «Кривые Безье»
е) «Формат BMP»
ж) «Формат GIF»
з) «Формат JPEG»
и) «Формат PNG»
к) «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 (без учёта палитры). Каково максимально возможное число цветов в палитре изображения?
14. Для хранения растрового изображения размером 128 × 128 пикселей отвели 4 килобайта памяти (без учёта палитры). 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).

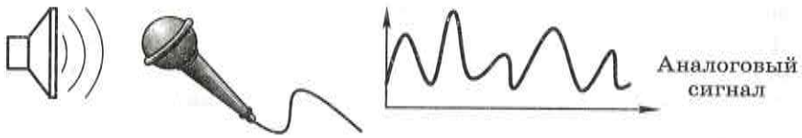


Рис. 2.24

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.

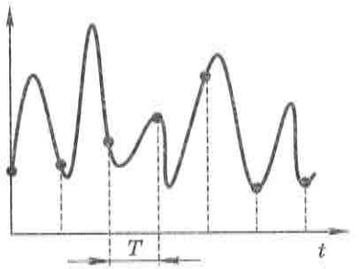


Рис. 2.25

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, однако восстановить точно исходный сигнал всё равно не удаётся, так как информация о значениях сигнала между моментами квантования была потеряна при оцифровке (см. рис. 2.26).

Для повышения качества звука, т. е. для большего соответствия между сигналом, принятым микрофоном, и сигналом, вы-

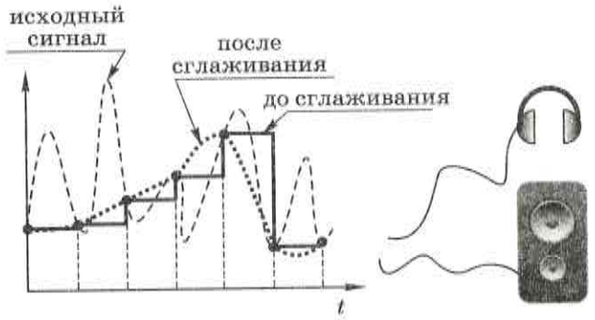


Рис. 2.26

веденным 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 кодировании? Ответ на этот вопрос во многом это 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. Это результат известен в радиотехнике как *теорема Котельникова—Шеннона*.

К сожалению, на практике всё несколько сложнее. Дело в том, что в реальных сигналах содержатся гармоники с очень высокими частотами, так что частота дискретизации, полученная с помощью теоремы Котельникова—Шеннона, будет также высока и объём файла — недопустимо велик. Однако средний человек слышит только звуки с частотами от 16 Гц до 20 кГц, поэтому все частоты выше 20 кГц можно «потерять» практически без ухудшения качества звука (человек не почувствует разницу!). Удвоив эту частоту (по теореме Котельникова—Шеннона), получаем оптимальную частоту дискретизации около 40 кГц, которая обеспечивает наилучшее качество, различимое на слух. Поэтому при высококачественном цифровом кодировании звука на компакт-дисках и в видеофильмах чаще всего используют частоты 44,1 и 48 кГц. Более низкие частоты дискретизации применяют тогда, когда важно всячески уменьшать объём звуковых данных (например, для трансляции радиопередач через Интернет), даже ценой ухудшения качества.

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).

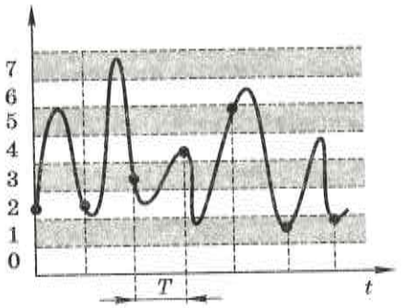


Рис. 2.27

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 (глубина кодирования) — это число битов, используемых 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; программа, выполняющая сжатие, называется **кодек** (от англ. *coder/decoder* — кодировщик/декодировщик).

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;
- **AAC** (англ. *Advanced Audio Coding*, файлы с расширениями aac, mp4, m4a и др.) — улучшенный формат кодирования звука, использующий сжатие данных и обеспечивающий более высокое качество, чем MP3; используется в магазине музыки iTunes Store фирмы Apple;
- **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 инструментов (можно использовать 128 мелодических и 47 ударных инструментов), громкость, тембр, время затухания каждой ноты и т. д. Фактически, это программа, предназначенная для проигрывания звуковой картой, в памяти которой хранятся образцы звуков реальных инструментов (*волновые таблицы*, англ. *wave tables*).

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

Звук, закодированный с помощью стандарта MIDI, хранится в файлах с расширением mid. 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.

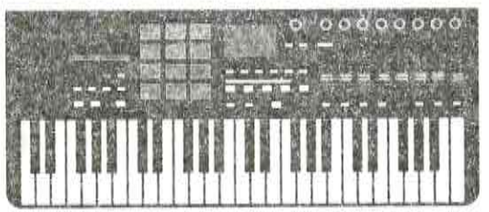


Рис. 2.28

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.

Существует так называемая трекерная музыка (от англ. *track* — дорожка), которая проигрывается с помощью сэмплов, хранящихся в том же файле, что и нотная запись. Файлы с трекерной музыкой (с расширениями mod, s3m, xm и др.) могут содержать несколько дорожек, звучащих одновременно.

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. 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. Для того чтобы ещё больше уменьшить объём файла, применяют *сжатие с потерями*, при котором теряются некоторые детали, несущественные для восприятия человеком. При очень сильном сжатии с потерями появляются заметные искажения (*артефакты*), например, становится явно видно, что изображение разбито на блоки размером 8 × 8 пикселей.

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, разработанный фирмой Microsoft для системы Windows; может использовать разные алгоритмы сжатия (кодеки);
- **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. Для каждого варианта вычислите объёмы звуковых файлов (без сжатия):

	1	2	3	4	5	6	7	8	9	10
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. Для каждого варианта вычислите время звучания записи (объёмы файлов приведены без учёта сжатия):

	1	2	3	4	5	6	7	8	9	10
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. Для каждого варианта вычислите глубину кодирования звука (объёмы файлов приведены без учёта сжатия):

	1	2	3	4	5	6	7	8	9	10
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. Для каждого варианта вычислите частоту дискретизации звука (объёмы файлов приведены без учёта сжатия):

	1	2	3	4	5	6	7	8	9	10
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?

	1	2	3	4	5	6	7	8	9	10
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 изображения
- 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 и шестнадцатеричная системы.
- Все типы данных в компьютерах кодируются в виде чисел, представленных в двоичной системе счисления.
- Символы хранятся в памяти компьютера в виде числовых кодов. Соответствие между символами и их кодами задаётся в таблицах кодировки. На практике чаще всего применяют 8-битные кодировки, позволяющие использовать 256 различных символов, и многобайтные кодировки стандарта UNICODE, содержащие полный набор символов, входящих во все известные языки.
- Существует два метода кодирования рисунков: растровое и векторное.
- При растровом кодировании изображение разбивается на пиксели. Качество растрового кодирования зависит от разрешения и глубины цвета. При растровом кодировании, как правило, происходит потеря информации. Растровый рисунок искажается, когда его размеры изменяются.
- При векторном кодировании рисунок представляется в виде набора геометрических фигур, которые задаются координатами своих узловых точек, а также цветами и стилями контура и заливки. Векторное кодирование чертежей, схем, карт и других изображений с чёткими границами объектов не приводит к потере информации. Векторный рисунок не искажается при изменении его размеров.
- Для кодирования звука применяют оцифровку и инструментальное кодирование. Качество кодирования определяется частотой дискретизации и глубиной кодирования.
- Кодирование видеофильмов сводится к кодированию изменяющегося изображения и звуковых дорожек. Изображение и звук должны проигрываться одновременно. Для уменьшения объёма файлов при кодировании видеоданных, как правило, используют сжатие.

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 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. Беги сюда!

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 двоичных 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: «Неверно, что сейчас идёт дождь».
НЕ B: «Неверно, что форточка открыта».
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**, и говорят, что они составляют **базис**.

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.

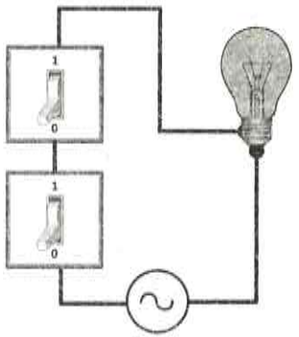


Рис. 3.2

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 `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-х 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 $A = B = 0$.

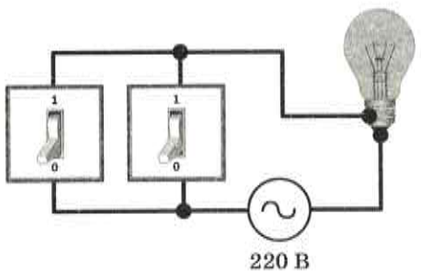


Рис. 3.4

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 похоже на обычное математическое сложение. Единственное отличие — в последней строке таблицы истинности: в арифметике 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. «Исключающее ИЛИ» в алгебре логики обозначается знаком $\oplus$, в языке Паскаль — словом `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	$\overline{A}$	$\overline{B}$	$\overline{A} \cdot B$	$A \cdot \overline{B}$	$\overline{A} \cdot B + A \cdot \overline{B}$
0	0	1	1	0	0	0
0	1	1	0	1	0	1
1	0	0	1	0	1	1
1	1	0	0	0	0	0

$A \oplus B$
0
1
1
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 A и B. 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 B, мы восстановим исходное значение. В этом смысле «исключающее ИЛИ» — обратимая операция (кроме неё обратима также операция «НЕ» — если применить её дважды, мы вернемся 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.

Эквиваленция

Эквиваленция (её также называют **«эквивалентность»**, **«равносильность»**, **«логическое равенство»**) — 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 эквиваленция — 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 эквиваленцию 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 их них можно записать произвольную логическую операцию. Например, операции «НЕ», «И» и «ИЛИ» (базовый набор, или *базис*) можно выразить через штрих Шеффера так:

$$\overline{A} = A \mid A, \quad A \cdot B = (A \mid B) \mid (A \mid B),$$
$$A + B = (A \mid A) \mid (B \mid B).$$

Эти формулы можно доказать через таблицы истинности. Поэтому одна операция «И-НЕ» тоже представляет собой базис. То же самое можно сказать и про операцию «ИЛИ-НЕ».

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) эквиваленция.

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.

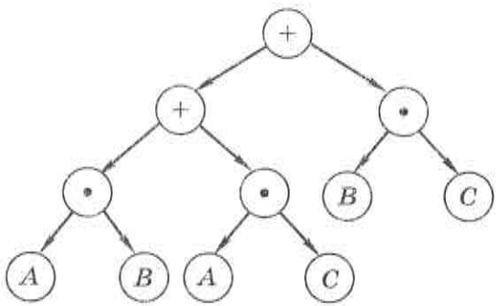


Рис. 3.12

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 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?

$$Z = (90 < X^2) \to (80 > (X+2)^2)$$

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

$$Z = (90 \geq X^2) \text{ или } (80 > (X+2)^2).$$

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 высказываний:

$$\overline{(A = B)}, \quad (A > B) \to (B > C), \quad (B > A) \to (C > B).$$

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. В этом случае импликация $(B > A) \to (C > B)$ превращается в истинное высказывание $0 \to 0$, так что 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, какой утюг и какой лоб?
22. Как выразить импликацию через операции «НЕ» и «ИЛИ»? Докажите полученное тождество.
23. Как выразить эквиваленцию через операции «НЕ», «И» и «ИЛИ»? Докажите полученное тождество.
24. Чем интересны операции «штрих Шеффера» и «стрелка Пирса»?
25. Что такое формализация?
26. В каком порядке выполняются действия в логических выражениях?

27. Что можно сделать для того, чтобы изменить естественный порядок действий?
28. Какие операции называются бинарными и унарными? Приведите примеры унарных и бинарных операций в математике.
29. Поясните разницу между терминами «логическое выражение» и «логическая функция».
30. Можно ли сказать, что таблица истинности однозначно определяет:
    а) логическое выражение;
    б) логическую функцию?
31. Что такое вычислимое логическое выражение?
32. Что такое тавтология? Противоречие? Приведите примеры.
33. 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) $\overline{\overline{(A \cdot C)} \cdot \overline{(B \cdot C)}}$;
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) $(\overline{A} \to \overline{B}) \to (A \to \overline{C})$;
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) $\overline{X} + \overline{Y} + \overline{Z}$.

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 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	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 трёх 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

$$((C < A) + (C < B)) \cdot \overline{((C + 1) < A)} \cdot \overline{((C + 1) < B)}.$$

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 её значение истинно, а вне его — ложно. Круги могут пересекаться. Области, в которых рассматриваемое логическое выражение истинно, закрашиваются каким-либо цветом. На рисунке 3.15 приведены диаграммы для простейших операций с одной и двумя переменными. Серым цветом залиты области, где рассматриваемое выражение равно единице.

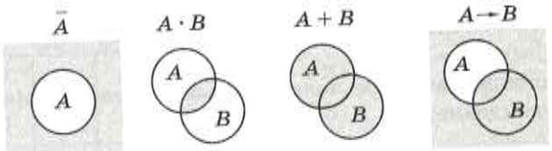
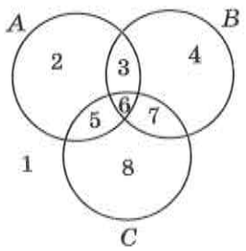


Рис. 3.15

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. 3.16 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. 3.16

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.

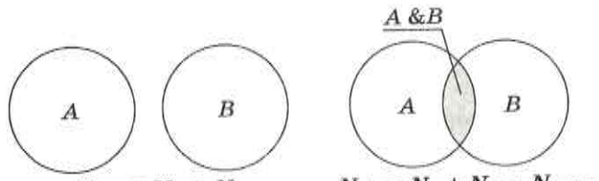
Диаграммы удобно применять для решения задач, в которых используются множества, например, множества страниц, полученных от поисковой системы в ответ на какой-то запрос. Рассмотрим следующую задачу.

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 (вариант *а* на рис. 3.17) или пересекаться (вариант *б*).



а *б*

Рис. 3.17

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. 3.17, *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 найдет этот сервер по запросу

(кошки | собаки) & лемуры?

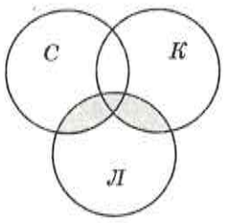


Рис. 3.18

Обозначим буквами *С*, *К* и *Л* области (группы сайтов), содержащие ключевые слова «собаки», «кошки» и «лемуры» соответственно (рис. 3.18). Построим диаграмму с тремя переменными и выделим интересующую область, которая соответствует запросу

(кошки | собаки) & лемуры.

На рисунке 3.18 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. 3.19.

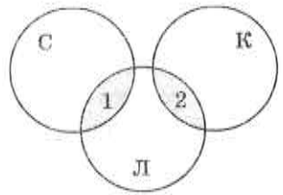


Рис. 3.19

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 найдет 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 найдет 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 найдет 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 найдет 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 найдет 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» (рис. 3.20).

Закон	Для «И»	Для операции «ИЛИ»
двойного отрицания	$\overline{\overline{A}} = A$	
исключённого третьего	$A \cdot \overline{A} = 0$	$A + \overline{A} = 1$
операции с константами	$A \cdot 1 = A,\ \ A \cdot 0 = 0$	$A + 1 = 1,\ \ A + 0 = A$
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	$\overline{A \cdot B} = \overline{A} + \overline{B}$	$\overline{A + B} = \overline{A} \cdot \overline{B}$

Рис. 3.20

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» заменяется 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, эквиваленцию 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

$$(\overline{(B+C)}\cdot A)\to(\overline{A}\cdot\overline{C}+D)=0.$$

Способ 1. 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 $B\cdot\overline{C}\cdot A=1$, 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.

Способ 2. 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 $A=1$, $B=C=D=0$.

Способ 3. Можно 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	повторяется
1	1	1	1	

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) $\overline{(A + \overline{B})} + \overline{(\overline{A} + B)} + A \cdot B$;
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) $A + \overline{B \cdot \overline{C}} + \overline{(A \to \overline{B \cdot C})}$.

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. 3.21).

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. 3.21

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 логическое 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 столбце X таблицы 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. 3.22).

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. 3.22

Отметим все строки, где $X = 1$, и для каждой из них построим выражение, истинное только для этой комбинации переменных (см. рис. 3.22). 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 в столбце X 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 высказываний невозможно доказать некоторые довольно простые утверждения. Рассмотрим такие высказывания:

«Все люди смертны».
«Сократ — человек».

Каждый из нас понимает, что если оба эти 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 «E», 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(Сократ) = 1. Поэтому с помощью таблицы истинности для импликации мы находим, что Q(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 записано 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: