

*Книга посвящается моей бабушке
Торгашовой Фаине Ивановне*

Предисловие

Предлагаемая читателям книга представляет собой руководство к решению задач по информатике. Круг рассматриваемых в ней вопросов достаточно широк.

В первой главе изучаются основные теоретические понятия: информация, система счисления, система передачи данных.

Во второй главе излагаются основы алгебры логики, даны примеры решения качественных логических задач, а также задач, связанных с анализом и синтезом переключательных и комбинационных схем, рассматриваются основы комбинаторики, теории множеств и теории игр.

Третья глава посвящена современным информационным технологиям. В ней рассматриваются электронные таблицы, файловые системы, информационные сети. Большое внимание в третьей главе также уделено вопросам моделирования, численным методам и решению оптимизационных задач.

В четвертой главе излагаются основы алгоритмизации и разобран ряд типовых задач.

Пятая глава представляет собой введение в программирование на языке Паскаль. В ней рассматриваются основные конструкции этого языка и примеры различных алгоритмов обработки данных.

После каждой главы даны задачи для самостоятельного решения. Всего таких задач — более 400, они приведены в порядке увеличения их сложности.

В разделе «Справочная информация» содержатся основные сведения о действиях со степенями, указаны свойства логарифма, даны таблицы сложения и умножения в некоторых системах счисления, проведено сопоставление встроенных функций электронных таблиц Microsoft Excel и OpenOffice.org Calc, а также даны для справки таблицы кодировки символов.

Безусловно, приведенная в данной книге теоретическая информация, а также примеры решений задач не являются исчерпываю-

щими в каждом из отдельных вопросов. Однако они дают достаточное представление о содержании основных задач по информатике и о наиболее распространенных способах их решения.

Книга будет полезной для школьников и студентов средних специальных учебных заведений, а также при самостоятельной подготовке к ЕГЭ по информатике.

Автор выражает благодарность Н. Н. Самылкиной за ценные советы, полученные при подготовке рукописи книги.

Глава 1

Информационные процессы и системы

1.1. Информация и ее кодирование

1.1.1. Понятие информации

Информация — это знания об объектах и явлениях окружающей среды, их параметрах, свойствах и состояниях. Получение информации об объектах уменьшает имеющуюся степень неопределенности, неполноты знаний о них. С информацией связаны *процессы передачи, хранения и обработки*, называемые *информационными процессами*.

Информация может существовать в различных *формах*. Для представления и передачи информации удобно использовать *знаковые системы*. **Знаковая система** включает в себя определенный набор знаков (**алфавит**) и правила работы с ними (**грамматика**).

Одним из примеров знаковой системы является **язык** — множество символов и совокупность правил, определяющих способы составления из этих символов осмысленных сообщений.

Языки делятся на *естественные* (русский, английский и т. д.) и *формальные* (азбука Морзе, языки программирования, язык математики и др.). Естественные языки складываются и развиваются стихийным образом, хотя и по своим законам. Правила оперирования со знаками в них сложны и неоднозначны; большая часть этих правил имеет исключения. Формальные же языки создаются для решения определенных задач и имеют ограниченный набор строгих правил.

1.1.2. Методы измерения количества информации

Единица измерения информации называется «**бит**». Термин «*бит*» представляет собой аббревиатуру от английского словосочетания «*binary digit*» («двоичная цифра»). Для измерения информации, наряду с битом, используются производные величины, указанные в табл. 1.1.

При измерении количества информации, содержащейся в некотором сообщении, обычно используются два метода: *вероятностный* и *алфавитный*.

Таблица 1.1

Единицы измерения информации

Название	Обозначение	Значение величины	
		в байтах	в битах
Байт	Б	2^0	2^3
Килобайт	КБ	2^{10}	2^{13}
Мегабайт	МБ	2^{20}	2^{23}
Гигабайт	ГБ	2^{30}	2^{33}
Терабайт	ТБ	2^{40}	2^{43}
Петабайт	ПБ	2^{50}	2^{53}
Эксабайт	ЭБ	2^{60}	2^{63}
Зеттабайт	ЗБ	2^{70}	2^{73}
Йоттабайт	ЙБ	2^{80}	2^{83}

Вероятностный подход

Вероятность — числовая характеристика возможности появления какого-либо события в определенных условиях.

Событием называется любой факт, который в результате опыта может произойти или не произойти. Вероятность появления некоторого события A вычисляется по формуле:

$$P_A = m / n, \quad (1.1)$$

где m — число благоприятных случаев (таких, при появлении которых обязательно должно совершиться событие A), а n — общее количество случаев. Вероятность любого события заключена в интервале от нуля до единицы: $0 \leq P_A \leq 1$.

П р и м е р. Подброшенная монета может упасть на «орла» или «решку». Количество возможных случаев здесь равно двум ($n = 2$). Упасть на «орла» монета может только в одном случае из двух ($m = 1$). Следовательно, вероятность того, что монета упадет на «орла», равна $P_o = m / n = 1 / 2$.

Зависимость между вероятностью появления события p и количеством информации I в сообщении о данном событии описывается формулой:

$$I = \log_2(1 / p), \quad (1.2)$$

т. е. чем меньше вероятность некоторого события, тем больше информации содержит сообщение об этом событии. (Здесь и ниже под со-

общением понимается некая последовательность чисел, с помощью которой закодирована информация. Способы кодирования текстовой, графической и другой информации будут рассмотрены позже.)

Алфавитный подход

Алфавитный метод измерения информации основан на том, что любое сообщение можно закодировать в виде конечной последовательности символов некоторого алфавита. Например, *двоичный алфавит* содержит два символа (0 и 1), а *троичный* — три символа (0, 1 и 2). Существуют алфавиты и с большим количеством символов. Так, алфавит *кодировки ASCII*, используемой для представления символьной информации, включает 256 элементов, а алфавит *кодировки UNICODE* — 65 536 элементов.

Количество различных сообщений N длиной в I символов, которые можно записать с помощью алфавита из q символов, оценивается с помощью формулы:

$$N = q^I. \quad (1.3)$$

Количество символов в алфавите q при этом называют **мощностью** (размером) **алфавита**:

$$q = \sqrt[I]{N}. \quad (1.4)$$

Количество символов I , используемых для записи N сообщений с помощью алфавита мощностью q символов, вычисляется по формуле:

$$I = \log_q N. \quad (1.5)$$

Например, если требуется закодировать 15 различных сообщений, то для этого можно использовать алфавит мощностью 2. В этом случае длина каждого сообщения будет составлять $I = \lceil \log_2 15 \rceil = 4$ символа (обозначение $\lceil \cdot \rceil$ означает округление в большую сторону, например: $\lceil 5.3 \rceil = 6$). А если для решения этой же задачи использовать алфавит мощностью 3, то длина сообщения будет меньше: $I = \lceil \log_3 15 \rceil = 3$ символа.

Отметим, что при решении задач, в которых требуется выполнить кодирование состояний некоторого объекта, каждое состояние соответствует одному какому-то символу алфавита.

При определении количества информации часто требуется вычислить заданную степень двойки. В этом вам поможет табл. 1.2.

Таблица 1.2

Степени двойки

Степень	Число	Степень	Число
0	$2^0 = 1$	10	$2^{10} = 1024$
1	$2^1 = 2$	11	$2^{11} = 2048$
2	$2^2 = 4$	12	$2^{12} = 4096$
3	$2^3 = 8$	13	$2^{13} = 8192$
4	$2^4 = 16$	14	$2^{14} = 16384$
5	$2^5 = 32$	15	$2^{15} = 32768$
6	$2^6 = 64$	16	$2^{16} = 65536$
7	$2^7 = 128$	17	$2^{17} = 131072$
8	$2^8 = 256$	18	$2^{18} = 262144$
9	$2^9 = 512$	19	$2^{19} = 542288$

Задача 1. Определить количество информации, которое содержится в сообщении о результате броска монеты.

Решение.

Возможно два равновероятных результата броска монеты: монета упала на «орла» или «решку». Следовательно, вероятности появления сообщений «монета упала на орла» и «монета упала на решку» одинаковы и равны $p = 1 / 2$. Вычислим количество информации в каждом сообщении о результате броска, используя формулу (1.2):

$$I = \log_2 (1 / (1 / 2)) = 1 \text{ бит.}$$

Ответ: 1 бит.

Задача 2. В барабане для розыгрыша лотереи находятся 16 пронумерованных шаров. Определите количество информации, содержащейся в сообщении о первом выпавшем номере.

Решение.

Запишем выражение для определения количества информации в сообщении, используя формулу (1.2) и зная, что вероятность выбора каждого шара $p = 1 / 16$:

$$I = \log_2 (1 / (1 / 16)) = 4 \text{ бита.}$$

Ответ: 4 бита.

Задача 3. Шахматная доска состоит из 64 полей: 8 столбцов на 8 строк. Какое минимальное количество битов потребуется для кодирования координат одного шахматного поля [1]?

Решение.

Вычислим количество битов для кодирования координат шахматного поля. В данной задаче событие соответствует выбору одного из 64 полей шахматной доски (т. е. всего существует 64 равновероятных события). Используя формулу (1.2), можно записать:

$$I = \log_2 (1 / (1 / 64)) = 6 \text{ бит.}$$

Ответ: 6 бит.

Задача 4. Световое табло состоит из лампочек, каждая из которых может находиться в двух состояниях («включено» или «выключено»). Какое наименьшее количество лампочек должно находиться на табло, чтобы с его помощью можно было передать 50 различных сигналов [4]?

Решение.

Лампочка может находиться в одном из двух состояний. Следовательно, для кодирования сигналов данного табло должен использоваться алфавит мощностью 2 символа ($q = 2$). Используя формулу (1.5), можно записать: $I = \log_2 50$, где I — количество лампочек (символов сообщения), требуемых для передачи 50 сигналов. Искомое I представляет собой ближайшее большее целое число для $\log_2 50$. Таким числом является 6, так как $6 = \log_2 64$ и $\log_2 64 > \log_2 50$.

Используя 6 лампочек, можно закодировать 64 различных сигнала. Следовательно, с помощью 6 лампочек можно закодировать и 50 сигналов. Если же использовать 5 лампочек ($5 = \log_2 32$), то с их помощью можно закодировать только 32 сигнала, что не удовлетворяет условию задачи.

Ответ: 6 лампочек.

Задача 5. Световое табло состоит из лампочек. Каждая лампочка может находиться в одном из трех состояний («включено», «выключено» или «мигает»). Какое наименьшее количество лампочек должно находиться на табло, чтобы с его помощью можно было передать 18 различных сигналов [5]?

Решение.

Лампочка может находиться в одном из трех состояний, поэтому для кодирования сигналов данного табло должен использоваться алфавит мощностью 3 символа ($q = 3$). Используя формулу (1.5), определим количество лампочек: $I = \log_3 18$, где I — количество лампочек, необходимых для передачи 18 сигналов. Искомое I представляет собой ближайшее большее целое число для $\log_3 18$. Таким числом является 3, так как $3 = \log_3 27$ и $\log_3 27 > \log_3 18$.

Ответ: 3 лампочки.

Задача 6. Какое наименьшее число символов должно быть в алфавите, чтобы при помощи всевозможных двухбуквенных слов, состоящих из символов данного алфавита, можно было передать не менее 11 различных сообщений?

Решение.

Используя формулу (1.3), запишем: $q^2 \geq 11$. Ближайшее целое, удовлетворяющее этому неравенству, — 4. Таким образом, алфавит должен содержать 4 символа.

Ответ: 4.

Задача 7. Сколько существует различных последовательностей из символов «плюс» и «минус» длиной ровно в пять символов [2]?

Решение.

Каждая из N последовательностей символов представляет собой набор «+» и «-» (0 и 1), поэтому для их кодирования должен использоваться алфавит мощностью 2 символа ($q = 2$). Следовательно, на основе формулы (1.3) можно записать: $N = 2^5 \Rightarrow N = 32$.

Ответ: 32.

Задача 8. В ящике находится 32 цветных карандаша, среди которых X — красного цвета. Наудачу вынимается один карандаш. Сообщение «извлечен не красный карандаш» несет 3 бита информации. Чему равно X ?

Решение.

Используя формулу (1.1), можно записать:

$$P_{нк} = (32 - X) / 32, \quad (1.6)$$

где $P_{нк}$ — вероятность выбора не красного карандаша.

На основе формулы (1.2) можно записать:

$$I = \log_2 (1 / P_{нк}). \quad (1.7)$$

Выразим $P_{нк}$ из формулы (1.7):

$$1 / P_{нк} = 2^I \Rightarrow P_{нк} = 1 / (2^I). \quad (1.8)$$

Зная, что $I = 3$, и подставляя (1.8) в (1.6), получим:

$$\begin{aligned} (32 - X) / 32 = 1 / (2^I) &\Rightarrow (32 - X) / 32 = 1 / (2^3) \Rightarrow \\ &\Rightarrow 32 - X = 4 \Rightarrow X = 28. \end{aligned}$$

Следовательно, в ящике находилось 28 красных карандашей.

Ответ: $X = 28$.

Задача 9. В двух вагонах товарного поезда находятся 32 генератора электрической энергии. Некоторые из них — исправные, некоторые — нет. Сообщение «сломанный генератор находится в первом вагоне» несет 4 бита информации. Сколько генераторов находится во втором вагоне?

Решение.

Зная, что в обоих вагонах находится 32 генератора, можно записать:

$$N_1 + N_2 = 32 \Rightarrow N_2 = 32 - N_1, \quad (1.9)$$

где N_1 — количество генераторов в первом вагоне, N_2 — во втором. Используя формулу (1.2), получим выражение для вычисления P_1 (вероятность того, что сломанный генератор находится в первом вагоне):

$$I = \log_2(1 / P_1) \Rightarrow 4 = \log_2(1 / P_1) \Rightarrow P_1 = 1 / 16. \quad (1.10)$$

На основе формул (1.1), (1.9) и (1.10) запишем:

$$P_1 = (32 - N_1) / 32 \Rightarrow (32 - N_1) / 32 = 1 / 16 \Rightarrow N_1 = 2.$$

Следовательно, из (1.9) получаем, что $N_2 = 32 - 2 = 30$.

Ответ: 30 генераторов.

Задача 10. В ящике находятся исправные и неисправные жесткие диски. Среди них 16 исправных. Сообщение о том, что из ящика достали неисправный диск, несет 1 бит информации. Сколько всего жестких дисков находится в ящике?

Решение.

В ящике находится 16 исправных дисков. Следовательно, $N_u + N_n = N \Rightarrow 16 + N_n = N$, где N — количество дисков в ящике, N_u — количество исправных дисков, N_n — количество неисправных дисков.

Используя формулу (1.1), запишем выражение для вычисления вероятности того, что из ящика достали неисправный диск:

$$P_n = N_n / (N_u + 16). \quad (1.11)$$

На основе (1.2) запишем:

$$I_n = \log_2(1 / P_n) \Rightarrow P_n = 1 / 2^{I_n} \Rightarrow P_n = 1 / 2^1 \Rightarrow P_n = 1 / 2. \quad (1.12)$$

Подставляя (1.12) в (1.11), получим: $1/2 = N_n / (N_u + 16) \Rightarrow 1 \cdot N_u = 16 \Rightarrow N_u = 16$. Таким образом, в ящике хранится $16 + 16 = 32$ жестких диска.

Ответ: 32 жестких диска.

Задача 11. Получено сообщение, информационный объем которого равен 32 битам. Чему равен этот объем в байтах?

Решение.

Выполним перевод битов в байты (1 байт = 8 бит): $32 / 8 = 4$.

Ответ: 4 байта.

Задача 12. Сколько мегабайт информации содержит сообщение объемом 2^{23} битов?

Решение.

Используя табл. 1.1, запишем:

$$2^{23} \text{ (бит)} = 2^{3+10+10} \text{ (бит)} = 2^{10+10} \text{ (байт)} = 2^{10} \text{ (кб)} = 1 \text{ (Мб)}.$$

Ответ: 1 Мб.

Задача 13. Сколько битов информации содержит сообщение объемом 4 мегабайта?

Решение.

Используя табл. 1.1, можно записать:

$$\begin{aligned} 4 \text{ (Мб)} &= 4 \cdot 2^{10} \text{ (кб)} = 4 \cdot 2^{10} \cdot 2^{10} \text{ (байт)} = \\ &= 2^2 \cdot 2^{10} \cdot 2^{10} \cdot 2^3 \text{ (бит)} = 2^{2+10+10+3} \text{ (бит)} = 2^{25} \text{ (бит)}. \end{aligned}$$

Ответ: 2^{25} бит.

1.2. Процесс передачи информации

Один из наиболее важных информационных процессов — **процесс передачи информации**. В нем участвуют различные устройства (см. рис. 1.1). Обмен информацией между *источником* и *получателем* выполняется по *каналу связи*. Каналы связи делятся на *симплексные* (информация по ним передается только в одну сторону, пример — телевидение) и *дуплексные* (в них возможна передача информации в обоих направлениях одновременно, примеры — телефон, локальная вычислительная сеть).

Кодирующее устройство предназначено для преобразования исходного сообщения источника к виду, удобному для передачи.

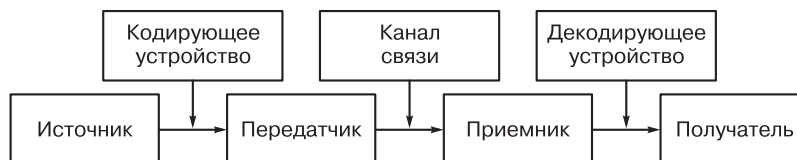


Рис. 1.1. Процесс обмена информацией

Декодировующее устройство предназначено для выполнения обратного преобразования кодированного сообщения в исходный вид.

Во время передачи информации может произойти ее частичная или полная потеря. Обычно это обусловлено ошибками в технической реализации канала связи, а также различными внешними воздействиями (помехами).

Каналы связи характеризуются *скоростью передачи информации, пропускной способностью и помехозащищенностью*. Для повышения помехозащищенности канала связи используются специальные методы передачи информации, уменьшающие влияние помех.

Скорость передачи информации измеряется в количестве битов, передаваемых за одну секунду («бит/с» (bps — bits per second)), т. е. в бодах (boud):

$$V = I / T, \quad (1.13)$$

где V — скорость передачи информации, I — объем передаваемой информации, а T — время передачи информации. Иногда в качестве единиц измерения скорости передачи информации используется байт в секунду («байт/с») и кратные ему единицы «кб/с» и «Мб/с».

Пропускная способность канала связи определяется максимальным объемом информации, передаваемым по нему в отсутствие помех:

$$I = V \cdot T. \quad (1.14)$$

Эта характеристика зависит от физических свойств канала. Пропускная способность показывает реальное количество битов в секунду, передаваемых по каналу связи.

Время передачи информации по каналу связи вычисляется по формуле:

$$T = I / V. \quad (1.15)$$

Задача 1. Скорость передачи данных через ADSL-соединение равна 128 000 бит/с. Передача файла через это соединение заняла 2 минуты. Определите размер файла в килобайтах.

Решение.

Вычислим объем переданных данных на основе формулы (1.14):

$$I = 128\,000 \cdot 60 \cdot 2 \text{ (бит)}.$$

Для перевода битов в килобайты используем табл. 1.1. Получим:

$$\begin{aligned}
 I &= 128\,000 \cdot 60 \cdot 2 / 8 \text{ (байт)} = \\
 &= 2^7 \cdot 1000 \cdot 60 \cdot 2 / (2^3 \cdot 1024) \text{ (кб)} = \\
 \{60 &= 2 \cdot 2 \cdot 15 = 2^2 \cdot 15, 1000 = 8 \cdot 125 = 2^3 \cdot 125, 1024 = 2^{10}\} = \\
 &= 2^7 \cdot 2^3 \cdot 125 \cdot 2^2 \cdot 15 \cdot 2 / (2^3 \cdot 2^{10}) = \\
 &= 2^{(1+2+7+3)} \cdot 15 \cdot 125 / 2^{13} = 15 \cdot 125 = 15 \cdot 250 = 3750 \text{ (кб)}.
 \end{aligned}$$

Ответ: 3750 кб.

Задача 2. Скорость передачи данных через ADSL-соединение равна 128 000 бит/с. Через данное соединение передают файл размером 625 килобайт. Определите время передачи файла в секундах [6].

Решение.

Переведем 625 килобайт в биты, используя табл. 1.1:

$$625 \text{ (кб)} = 625 \cdot 2^{13} \text{ (бит)}.$$

Используя формулу (1.15), запишем:

$$\begin{aligned}
 T &= 625 \cdot 2^{13} \text{ (бит)} / 128\,000 \text{ (бит/с)} = 625 \cdot 2^{13} / 128 \cdot 1000 \text{ (с)} = \\
 &= 625 \cdot 2^6 / 1000 \text{ (с)} = 40\,000 / 1000 = 40 \text{ (с)}.
 \end{aligned}$$

Ответ: 40 секунд.

Задача 3. У Васи есть доступ к Интернет по высокоскоростному одностороннему радиоканалу, обеспечивающему скорость получения им информации 256 Кбит/с. У Пети нет скоростного доступа в Интернет, но есть возможность получать информацию от Васи по низкоскоростному телефонному каналу со средней скоростью 32 Кбит/с. Петя договорился с Васей, что тот будет скачивать для него данные объемом 5 Мбайт по высокоскоростному каналу и ретранслировать их Пете по низкоскоростному каналу.

Компьютер Васи может начать ретрансляцию данных не раньше, чем им будут получены первые 512 Кбайт этих данных. Каков минимально возможный промежуток времени (в секундах) с момента начала скачивания Васей данных до полного их получения Петей [7]?

Решение.

Время, потраченное на передачу информации от Васи к Пете, состоит из двух частей:

- 1) время на получение первых 512 килобайт Васей: t_1 ;
- 2) время на передачу 5 мегабайт от Васи к Пете: t_2 .

Таким образом, минимально возможный промежуток времени с момента скачивания Васей данных до полного их получения Петей составляет $t = t_1 + t_2$.

Переведем из Кбит/с в кб/с скорость получения данных Васей из Интернета:

$$256 \text{ (Кбит/с)} = 256 / 8 \text{ (кб/с)} = 32 \text{ (кб/с)}.$$

Вычислим t_1 , используя формулу:

$$t_1 = 512 / 32 = 2^9 / 2^5 = 2^4 = 16 \text{ с.}$$

Переведем скорость передачи информации от Васи к Пете из Кбит/с в кб/с:

$$32 \text{ (Кбит/с)} = 32 / 8 \text{ (кб/с)} = 4 \text{ (кб/с)}.$$

Вычислим t_2 :

$$t_2 = 5 \cdot 1024 / 4 = 5 \cdot 2^{10} / 2^2 = 5 \cdot 2^8 = 5 \cdot 256 = 1280 \text{ с,}$$

$$t_2 = 16 + 1280 = 1296 \text{ с.}$$

Ответ: 1296 с.

1.3. Представление числовой информации

Представление чисел в ЭВМ основано на использовании понятия «система счисления».

Система счисления — это способ представления чисел и соответствующие ему правила действия над числами. Знаки, используемые при записи чисел, называются *цифрами*.

Разнообразные системы счисления можно разделить на непозиционные и позиционные. В *непозиционных системах счисления* от положения цифры в записи числа не зависит величина, которую эта цифра обозначает. В *позиционных системах счисления*, наоборот, величина, обозначаемая цифрой в записи числа, зависит от ее позиции в числе. Количество используемых цифр q при этом называется *основанием позиционной системы счисления*.

Система счисления, применяемая в современной математике, является *позиционной десятичной системой* ($q = 10$). Запись чисел в ней производится с помощью десяти цифр: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. Позиционный характер этой системы счисления легко понять на примере любого многозначного числа. Например, в числе 333 первая тройка означает «три сотни», вторая — «три десятка», а третья — «три единицы». Отдельную позицию в представлении числа принято называть *разрядом*, а номер позиции — *номером разряда*. Количество разрядов в записи числа называется *разрядностью* и совпадает с его длиной.

Для записи чисел в позиционной системе счисления с основанием q нужен алфавит из q цифр. Обычно при $q < 10$ используют первые q арабских цифр, а при $q > 10$ к десяти арабским цифрам добавляют буквы. В табл. 1.3 приведены примеры алфавитов нескольких таких систем счисления.

Таблица 1.3

Основания и алфавиты систем счисления

Основание	Название	Алфавит
$q = 2$	двоичная	0 1
$q = 3$	троичная	0 1 2
$q = 8$	восьмеричная	0 1 2 3 4 5 6 7
$q = 16$	шестнадцатеричная	0 1 2 3 4 5 6 7 8 9 A B C D E F

В системе счисления с основанием q (q -ичная система счисления) единицами разрядов служат последовательные степени числа q , а q единиц какого-либо разряда образуют единицу следующего разряда. Для записи числа в q -ичной системе счисления требуется q различных знаков (цифр), изображающих числа: 0, 1, ..., $q - 1$. Запись числа q в q -ичной системе счисления имеет вид **10**, т. е. данное число состоит из одного десятка и нуля единиц.

Основание системы счисления, в которой записано число, обычно записывают после числа в виде нижнего индекса, например: 101101_2 , 3671_8 , $3B8F_{16}$.

1.3.1. Перевод чисел между системами счисления

Перевод числа, записанного в системе счисления с основанием q , в десятичную систему счисления основан на использовании *развернутой формы записи числа*:

$$A_q = \pm (a_{n-1}q^{n-1} + a_{n-2}q^{n-2} + \dots + a_0q^0 + a_{-1}q^{-1} + a_{-2}q^{-2} + \dots + a_{-m}q^{-m}), \quad (1.16)$$

где A_q — само данное число, q — основание системы счисления, a_i — цифры q -ичной системы счисления, n — количество разрядов целой части числа, m — количество разрядов дробной части числа.

Примеры. Развернутая форма чисел $24,3891_{10}$, $101,11_2$ и $F1,FD_{16}$ имеет вид:

- 1) $24,3891_{10} = 2 \cdot 10^1 + 4 \cdot 10^0 + 3 \cdot 10^{-1} + 8 \cdot 10^{-2} + 9 \cdot 10^{-3} + 1 \cdot 10^{-4}$;
- 2) $101,11_2 = 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 1 \cdot 2^{-1} + 1 \cdot 2^{-2}$;
- 3) $F1,FD_{16} = F \cdot 16^1 + 1 \cdot 16^0 + F \cdot 16^{-1}$.

Таким образом, для перевода числа A_q в десятичную систему счисления необходимо:

- 1) записать развернутую форму числа A_q ;
- 2) представить все слагаемые развернутой формы в десятичной системе счисления;
- 3) вычислить полученное выражение по правилам десятичной арифметики.

При переводе чисел между системами счисления для приведения получаемых цифр в соответствии с алфавитом конечной системы счисления можно использовать табл. 1.4 и 1.5.

Таблица 1.4

Соответствие между двоичными и шестнадцатеричными числами

2	0000	0001	0010	0011	0100	0101	0110	0111
16	0	1	2	3	4	5	6	7
10	0	1	2	3	4	5	6	7

2	1000	1001	1010	1011	1100	1101	1110	1111
16	8	9	A	B	C	D	E	F
10	8	9	10	11	12	13	14	15

Таблица 1.5

Соответствие между двоичными и восьмеричными числами

2	000	001	010	011	100	101	110	111
8	0	1	2	3	4	5	6	7
10	0	1	2	3	4	5	6	7

Часто при решении различных задач бывает необходимо вычислить количество разрядов десятичного числа в некоторой системе счисления. Сделать это можно двумя способами.

1. Можно перевести число A в указанную систему счисления, тогда количество цифр полученного числа соответствует искомому количеству разрядов. Однако в ряде случаев этот способ достаточно трудоемок.

2. Можно найти такие показатели степени m и n , чтобы выполнялось неравенство: $q^m < A < q^n$ (где $m < n$). Тогда n — количество разрядов числа A в q -ичной системе счисления. При поиске m и n в двоичной системе счисления удобно пользоваться табл. 1.2. Например, для числа 1123 можно записать неравенство: $2^{10} (1024) < 1123 < 2^{11} (2048)$.

Задача 1. Перевести числа 201_3 , $F1.F_{16}$ в десятичную систему счисления.

Решение.

Перевод числа 201_3 . Запишем развернутую форму числа:

$$201_3 = 2 \cdot 3^2 + 0 \cdot 3^1 + 1 \cdot 3^0.$$

Все слагаемые в этой развернутой форме представлены в десятичной системе счисления, поэтому никаких замен выполнять не нужно. Получаем:

$$2 \cdot 3^2 + 0 \cdot 3^1 + 1 \cdot 3^0 = 2 \cdot 9 + 0 \cdot 3 + 1 \cdot 1 = 19_{10}.$$

Перевод числа $F1.F_{16}$. Развернутая форма числа:

$$F1.F_{16} = F \cdot 16^1 + 1 \cdot 16^0 + F \cdot 16^{-1}.$$

В этой развернутой форме есть слагаемые, представленные в шестнадцатеричной системе счисления. Выполняем их замену:

$$F \cdot 16^1 + 1 \cdot 16^0 + F \cdot 16^{-1} = 15 \cdot 16^1 + 1 \cdot 16^0 + 15 \cdot 16^{-1}.$$

Получаем:

$$15 \cdot 16^1 + 1 \cdot 16^0 + 15 \cdot 16^{-1} = 15 \cdot 16 + 1 \cdot 1 + 15 \cdot 0,0625 = 241,9375_{10}.$$

Ответ: $201_3 = 19_{10}$; $F1.FD_{16} = 241,9375_{10}$.

Перевод целого десятичного числа A_{10} в систему счисления с основанием q

1. Выполняется последовательное *деление* числа A_{10} и получаемых частных на основание новой системы счисления q до тех пор, пока не будет получено частное, меньшее делителя (деление выполняется на основе правил десятичной арифметики).

2. Полученные остатки от деления приводятся в соответствие с алфавитом новой системы счисления.

3. Результирующее число составляется из последнего частного и остатков, записанных в обратном порядке.

Задача 2. Перевести число 37_{10} в двоичную систему счисления.

Решение.

Выполним деление числа 37 на 2 (жирным выделены цифры, участвующие в результирующем числе).

Полученные остатки находятся в соответствии с алфавитом двоичной системы счисления. Получаем: $37_{10} = 100101_2$.

Ответ: $37_{10} = 100101_2$.

$$\begin{array}{r}
 37 \overline{) 2} \\
 \underline{36} \quad 18 \quad 2 \\
 1 \quad 18 \quad 9 \quad 2 \\
 \quad 0 \quad 8 \quad 4 \quad 2 \\
 \quad \quad 1 \quad 4 \quad 2 \quad 2 \\
 \quad \quad \quad 0 \quad 2 \quad 1 \\
 \quad \quad \quad \quad 0
 \end{array}$$

Задача 3. Перевести число 335_{10} в шестнадцатеричную систему.

Решение.

Выполним деление числа 335 на 16:

$$\begin{array}{r|l} 335 & 16 \\ 320 & 20 \\ \hline 15 & 16 \\ \hline & 4 \end{array}$$

Полученные остатки приведем в соответствие с алфавитом шестнадцатеричной системы счисления: $1_{10} = 1_{16}$, $4_{10} = 4_{16}$, $15_{10} = F_{16}$. Получим: $335_{10} = 14F_{16}$.

Ответ: $335_{10} = 14F_{16}$.

Перевод дробного десятичного числа A_{10} в систему счисления q

1. Выполняется последовательное *умножение* числа A_{10} и получаемых дробных частей произведений на основание новой системы счисления q до тех пор, пока дробная часть произведения не станет равной нулю или не будет достигнута требуемая точность представления числа в новой системе счисления.

2. Полученные целые числа произведений приводятся в соответствие с алфавитом новой системы счисления.

3. Составляется дробная часть числа в новой системе счисления начиная с целой части первого произведения.

Задача 4. Перевести число $0,1875_{10}$ в двоичную систему счисления.

Решение.

Выполним умножение числа 0,1875 на 2 (в таблице жирным выделены цифры, участвующие в результирующем числе; в скобках указан номер цифры в результирующем числе):

Действие	Результат
$0,1875 \cdot 2 =$	0,3750 (1)
$0,3750 \cdot 2 =$	0,7500 (2)
$0,7500 \cdot 2 =$	1,5000 (3)
$0,5000 \cdot 2 =$	1,0000 (4)

Полученные целые части произведений находятся в соответствии с алфавитом новой системы счисления. Поэтому можно записать: $0,1875_{10} = 0,0011_2$.

Ответ: $0,1875_{10} = 0,0011_2$.

Задача 5. Перевести число $0,37245_{10}$ в восьмеричную систему счисления.

Решение.

Выполним умножение числа $0,37245$ на 8:

Действие	Результат	Действие	Результат
$0,37245 \cdot 8 =$	2,9796 (1)	$0,55520 \cdot 8 =$	4,4416 (5)
$0,97960 \cdot 8 =$	7,8368 (2)	$0,44160 \cdot 8 =$	3,5328 (6)
$0,83680 \cdot 8 =$	6,6944 (3)	$0,53280 \cdot 8 =$	4,2624 (7)
$0,69440 \cdot 8 =$	5,5552 (4)	$0,26240 \cdot 8 =$	2,0992 (8)

Полученные целые части произведений находятся в соответствии с алфавитом новой системы счисления. Поэтому можно записать: $0,37245_{10} = 0,27654342_8$. Ответ получен с точностью 8 цифр после запятой.

Ответ: $0,37245_{10} = 0,27654342_8$.

Перевод смешанного числа A_{10} в систему счисления q

Перевод смешанных чисел, содержащих целую и дробную части, осуществляется в два этапа. Целая и дробная части исходного числа переводятся отдельно, а в итоговой записи числа в новой системе счисления целая часть отделяется от дробной части запятой. Например, учитывая рассмотренные выше задачи, можно получить двоичное представление числа $37,1875_{10}$:

$$37,1875_{10} = 100101,0011_2.$$

1.3.2. Быстрый перевод чисел между системами счисления

Быстрый перевод чисел между системами счисления, основания которых кратны степени двойки (2, 4, 8, 16 и т. д.), можно выполнять без использования операций деления и умножения. Данный способ основан на том, что каждой цифре числа в системе счисления, основание которой q кратно степени двойки, соответствует число, состоящее из I ($q = 2^I$) цифр в двоичной системе. Например, одной цифре числа в шестнадцатеричной системе соответствуют 4 цифры ($16 = 2^4$) числа в двоичной системе, а каждой цифре числа в восьмеричной системе счисления соответствуют 3 цифры ($8 = 2^3$) в двоичной системе.

Алгоритм перевода числа A_2 из двоичной в шестнадцатеричную (восьмеричную) систему счисления:

- 1) дополнить число A_2 незначащими нулями так, чтобы его можно было разбить на целое количество групп цифр по четыре (по три) цифры в группе;
- 2) разбить цифры числа на группы по четыре (по три) цифры в группе;
- 3) найти соответствие каждой группе цифр числа A_2 цифре из шестнадцатеричной (восьмеричной) системы счисления (см. табл. 1.4 и табл. 1.5);
- 4) полученное число представляет собой искомый результат перевода.

Перевод числа A из шестнадцатеричной (восьмеричной) системы счисления в двоичную заключается в представлении каждой цифры числа ее двоичным кодом на основе табл. 1.4 и табл. 1.5.

Примечания.

1. Указанным способом нельзя переводить числа между двоичной и десятичной, между восьмеричной и десятичной и между шестнадцатеричной и десятичной системами счисления.
2. Для перевода чисел между шестнадцатеричной и восьмеричной системами счисления в качестве промежуточной можно использовать двоичную систему.

Задача 6. Определите, чему равно восьмеричное число $0,2(1)_8$ в системе счисления по основанию 16.

Решение.

Запишем периодическую дробь в двоичной системе счисления, учитывая, что одному разряду в восьмеричной системе счисления соответствуют три разряда в двоичной системе счисления. Повторим периодическую часть в восьмеричной системе счисления столько раз, сколько будет достаточно для выделения периодической части в двоичной системе:

$$\begin{aligned} 0,2(1)_8 &= 0,0100\ 0100\ 1001\ 0010\ 0100\ 1001\ 0010\ 0100\ 1001_2 = \\ &= 0,4\ 4\ 9\ 2\ 4\ 9\ 2\ 4\ 9_{16}. \end{aligned}$$

Получим периодическую дробь: $0,4(492)_{16}$.

Ответ: $0,4(492)_{16}$.

Задача 7. Перевести число $DD1_{16}$ в двоичную систему счисления.

Решение.

Переведем $DD1_{16}$ в двоичную систему счисления, используя табл. 1.4 и табл. 1.5:

$$DD1_{16} = 1101\ 1101\ 0001_2 = 110111010001_2.$$

Ответ: 110111010001_2 .

Задача 8. Перевести число 11010001_2 в восьмеричную систему счисления.

Решение.

Заданное число необходимо перевести в восьмеричную систему счисления. Дополним его нулями так, чтобы количество цифр стало кратно трем: $11010001_2 = 011010001_2$.

Разобьем цифры числа на группы по три: $011\ 010\ 001_2$.

Выполним перевод каждой группы цифр в восьмеричную цифру, используя табл. 1.5:

$$011 = 3, 010 = 2, 001 = 1.$$

Получаем: $011\ 010\ 001_2 = 321_8$.

Ответ: 321_8 .

Задача 9. Дано $a = D7_{16}$, $b = 331_8$. Определите, выполняется ли неравенство $D7_{16} < 11011000_2 < 331_8$ [6].

Решение.

Получим двоичное представление чисел $a = D7_{16}$ и $b = 331_8$:

$$a = 11010111_2, b = 011011001_2.$$

Заметим, что числа a и b начинаются одинаково: $a = 1101\ 0111$, $b = 1101\ 1001$ (незначащий нуль в представлении числа b опущен). Поэтому для сравнения чисел достаточно выполнять анализ только последних четырех цифр.

Для удобства сравнения можно перевести числа из двоичной системы счисления в десятичную: $a = 7_{10} = 0111_2$; $b = 9_{10} = 1001_2$.

Следовательно, единственным числом, которое удовлетворяет приведенному неравенству, является число, оканчивающееся на 1000, т. е. число 11011000_2 .

Ответ: неравенство выполняется.

Задача 10. Как представлено число 27_{10} в двоичной системе счисления?

Решение.

Выполним перевод числа 27_{10} в двоичную систему счисления, разделив 27 на 2:

$$\begin{array}{r|l}
 27 & 2 \\
 \hline
 26 & 13 \\
 \hline
 1 & 12 \\
 & 1 \\
 & 6 \\
 & 6 \\
 & 0 \\
 & 3 \\
 & 2 \\
 & 1
 \end{array}$$

Ответ: $27_{10} = 11011_2$.

Решение.

Выполним деление числа 29_{10} на 2:

$$\begin{array}{r|l}
 29 & 2 \\
 \hline
 28 & 14 \quad 2 \\
 \hline
 1 & 14 \quad 7 \quad 2 \\
 & 0 \quad 6 \quad 3 \quad 2 \\
 & & 1 \quad 2 \quad 1 \\
 & & & 1
 \end{array}$$

Ответ: 1.

Решение.

$$\begin{array}{r|rr} 25 & 4 & \\ 24 & 6 & 4 \\ \hline 1 & 4 & 1 \\ & \underline{2} & \end{array}$$

Зная, что алфавит системы счисления с основанием 4 содержит 4 цифры (0, 1, 2, 3), запишем все трехразрядные числа в данной системе счисления, оканчивающиеся на 11: $011_4, 111_4, 211_4, 311_4$.

Переведем указанные числа в десятичную систему счисления и выберем те из них, которые не превосходят 25:

$$1) 011_4 = 0 \cdot 4^2 + 1 \cdot 4^1 + 1 \cdot 4^0 = 4 + 1 = 5_{10};$$

$$2) 111_4 = 1 \cdot 4^2 + 1 \cdot 4^1 + 1 \cdot 4^0 = 16 + 4 + 1 = 21_{10};$$

$$3) 211_4 = 2 \cdot 4^2 + 1 \cdot 4^1 + 1 \cdot 4^0 = 2 \cdot 16 + 4 + 1 = 37_{10};$$

$$4) 311_4 = 3 \cdot 4^2 + 1 \cdot 4^1 + 1 \cdot 4^0 = 3 \cdot 16 + 4 + 1 = 53_{10}.$$

Условиям задачи отвечают числа 5_{10} и 21_{10} . (При решении задачи можно прекратить вычисления при получении числа, большего 25.)

Ответ: 5, 21.

Задача 13. Найдите основание системы счисления, в которой верно следующее равенство: $4202_p - 2221_p = 1431_p$.

Решение.

Запишем развернутую форму чисел, участвующих в равенстве:

$$\begin{aligned} (4 \cdot p^3 + 2 \cdot p^2 + 0 \cdot p^1 + 2 \cdot p^0) - (2 \cdot p^3 + 2 \cdot p^2 + 2 \cdot p^1 + 1 \cdot p^0) = \\ = (1 \cdot p^3 + 4 \cdot p^2 + 3 \cdot p^1 + 1 \cdot p^0). \end{aligned}$$

Преобразуем уравнение:

$$\begin{aligned} 4 \cdot p^3 + 2 \cdot p^2 + 2 - 2 \cdot p^3 - 2 \cdot p^2 - 2 \cdot p - 1 - p^3 - 4 \cdot p^2 - 3 \cdot p - 1 = \\ = p^3 - 4 \cdot p^2 - 5 \cdot p = 0. \end{aligned}$$

Вынесем p за скобку: $p \cdot (p^2 - 4 \cdot p - 5) = 0$. Следовательно, $p = 0$ или $p^2 - 4 \cdot p - 5 = 0$. При этом решение $p = 0$ не удовлетворяет условию, что основание системы счисления должно быть больше 0.

Рассмотрим уравнение $p^2 - 4 \cdot p - 5 = 0$. Оно имеет два решения: $p = 5$ или $p = -1$. Основание системы счисления отрицательным быть не может. Значит, указанное равенство верно в системе счисления с основанием 5.

Ответ: 5.

Задача 14. В системе счисления с некоторым основанием число 17 записывается в виде 101. Укажите это основание [3].

Решение.

Запишем развернутую форму числа 101, используя формулу (1.16) и учитывая, что основание системы счисления неизвестно:

$$1 \cdot x^2 + 0 \cdot x^1 + 1 \cdot x^0 = 17.$$

Решая уравнение, получаем: $x^2 = 16 \Rightarrow x = 4$. Поэтому число 17 представляется в виде 101 в системе счисления с основанием 4.

Ответ: 4.

Задача 15. Укажите наименьшее основание системы счисления, в которой запись числа 21 трехзначна.

Решение.

Десятичная запись числа 21 — двухзначное число. Поэтому искомое основание системы счисления меньше 10, т. е. $q < 10$. Вычислим количество разрядов, которое занимает число 21 в системах счисления с основанием меньше 10, начиная с двоичной:

1) $q = 2$: $2^5(32) < 21 < 2^4(16)$: запись числа 21 при $q = 2$ содержит 5 разрядов;

2) $q = 3$: $3^2(9) < 21 < 3^3(27)$: запись числа 21 при $q = 3$ содержит 3 разряда.

Получим, что запись числа 21_{10} трехзначна в троичной системе счисления.

Ответ: 3.

Задача 16. Определите наименьшее основание позиционной системы счисления x , при котором $104_x = 646_y$.

Решение.

Основание системы счисления больше значения максимальной цифры данной системы. Поэтому на значения x и y накладываются следующие ограничения: $x \geq 5$, $y \geq 7$.

Переведем обе части уравнения в десятичную систему счисления:

$$x^2 + 4 = 6 \cdot y^2 + 4 \cdot y + 6 \Rightarrow x^2 = 6 \cdot y^2 + 4 \cdot y + 2 \Rightarrow x = \sqrt{6 \cdot y^2 + 4 \cdot y + 2}.$$

Определим значения x и y , являющиеся решением задачи. Для этого будем перебирать значения y , начиная с 7, и подставлять их в уравнение, определяя x (причем x должен быть целым):

$$y = 7, x = \sqrt{324} = 18. \text{ Решение найдено за один шаг.}$$

Ответ: 18.

Задача 17. Укажите через запятую в порядке возрастания все основания систем счисления, в которых запись числа 23 оканчивается на 2 [2].

Решение.

Используя (1.16), запишем: $23 = x \cdot y^1 + 2 \cdot y^0$, $x \cdot y + 2 = 23$. Это означает, что десятичное число 23 в системе счисления с основанием y содержит x десятков и две единицы. Таким образом, для решения задачи необходимо найти все целые числа x и y , которые удовлетворяют приведенному равенству.

x	y	Проверка
21	1	$21 \cdot 1 + 2 = 23$
1	21	$1 \cdot 21 + 2 = 23$
3	7	$3 \cdot 7 + 2 = 23$
7	3	$7 \cdot 3 + 2 = 23$

Внимание! Записи « $3 \cdot 7 + 2 = 23$ » и « $7 \cdot 3 + 2 = 23$ » являются различными: в первом случае 3 обозначает количество десятков, а 7 — основание системы счисления; во втором случае 7 обозначает количество десятков, а 3 — основание системы счисления.

Получим возможные основания систем счисления: 1, 3, 7, 21. Основание системы должно быть строго больше количества единиц, т. е. больше 2. Следовательно, число 23 оканчивается на 2 в системах счисления с основанием 3, 7 и 21.

Ответ: 3, 7, 21.

Задача 18. Число $x = 121$, рассматриваемое в системе счисления с основанием p ($2 < p < 10$), представляет собой наименьшее число, кратное десятичному числу 36. Найдите основание p системы счисления, не перебирая все возможные значения p .

Решение.

Число x является кратным числу y , если x делится на y без остатка. Учитывая это и записывая развернутую форму числа 121_x , получим следующее выражение:

$$(p^2 + 2 \cdot p + 1) / 36 = k, \text{ где } k \text{ — целое число } (k \geq 1). \quad (1.17)$$

Из условия задачи можно сделать вывод, что результат деления числа x на число 36 равен 1 ($k = 1$), так как число x представляет собой наименьшее число, кратное 36 (большие числа, кратные 36, будут давать большее k).

Подставим $k = 1$ в формулу (1.17) и решим полученное уравнение:

$$(p^2 + 2 \cdot p + 1) / 36 = k; \Rightarrow p^2 + 2 \cdot p + 1 = 36; \Rightarrow p^2 + 2 \cdot p - 35 = 0;$$

$$D = 2^2 - 4 \cdot 1 \cdot (-35) = 4 + 140 = 144; \Rightarrow \sqrt{D} = 12.$$

$$p_1 = (-2 + 12) / (2 \cdot 1), \quad p_2 = (-2 - 12) / (2 \cdot 1); \Rightarrow p_1 = 5, p_2 = -7.$$

Получаем, что число 121_x записано в системе счисления с основанием 5 (число -7 решением не является, так как отрицательное число основанием системы счисления быть не может).

Проверка:

$$1 \cdot 5^2 + 2 \cdot 5^1 + 1 \cdot 5^0 = 25 + 2 \cdot 5 + 1 = 36.$$

Ответ: 5.

Задача 19. Шестнадцатеричное четырехзначное число заканчивается цифрой 7. Первую цифру переставили в конец числа. Полученное число оказалось на $B09A_{16}$ больше исходного. Определите, чему равно исходное число, записанное в системе счисления по основанию 16.

Решение.

Обозначим цифры числа через X , Y и Z . Составим уравнение:
 $XYZ7 + B09A = YZ7X$:

$$\begin{array}{r} XYZ7 \\ + B09A \\ \hline YZ7X \end{array}$$

Определим значения X , Y и Z :

1) $7 + A = 11$, $X = 1$, единица переносится в старший разряд:

$$\begin{array}{r} 1 \\ XYZ7 \\ + B09A \\ \hline YZ71 \end{array}$$

2) существует несколько возможных значений суммы $Z + 9 + 1 = 7$, 17 или 27 , т. е. существует несколько возможных значений очередного переноса в старший разряд. Рассмотрим каждый из них:

а) $Z + 9 + 1$ не может быть равно 7_{16} , так как в сумме присутствует цифра 9, которая больше 7;

б) $Z + 9 + 1$ не может быть равно 27_{16} , так как максимальное значение Z равно F_{16} , но тогда $F + 9 + 1 = 19_{16}$ (а $19_{16} < 27_{16}$), т. е. при максимально возможном значении Z будет получено число, меньшее 27_{16} ;

в) следовательно, $Z + 9 + 1 = 17$, т. е. $Z = D_{16}$ (и единица переносится в старший разряд):

$$\begin{array}{r} 1 \\ XCD7 \\ + B09A \\ \hline YZ71 \end{array}$$

3) $Y + 0 + 1 = D \Rightarrow Y = C$;

4) $X + B = Y \Rightarrow 1 + B = C$.

Вывод: исходное число $XYZ7 = 1CD7$.

Ответ: $1CD7$.

Задача 20. Не переводя непосредственным делением «в столбик» десятичное число 4097 в двоичную систему, определите количество нулей в его двоичном представлении.

Решение.

Число 4097 можно представить в виде $4097 = 4096 (= 2^{12}) + 1$. Количество нулей в числе, которое представляет собой степень двойки, равно самой данной степени, т. е. в число 2^{12} входит 12 нулей. Поэтому $4096_{10} = 1\ 0000\ 0000\ 0000_2$. Прибавим к данному числу единицу:

$1\ 0000\ 0000\ 0000_2 + 1_2 = 1\ 0000\ 0000\ 0001_2$. Получаем, что в двоичном представлении числа 4097 содержится 11 нулей.

Ответ: 11.

Задача 21. Укажите, сколько раз используется цифра 0 при записи чисел 13, 14, 15, ..., 19, 20 в системе счисления с основанием 3.

Решение.

Переведем число 13 в систему счисления с основанием 3: $13_{10} = 111_3$. Запишем таблицу, в которой каждое последующее число получается из предыдущего прибавлением к его младшему разряду единицы:

Десятичное	13	14	15	16	17	18	19	20
Троичное	111	112	120	121	122	200	201	202
Кол-во нулей	0	0	1	0	0	2	1	1

Таким образом, при записи указанных чисел цифра 0 используется 5 раз.

Ответ: 5.

Задача 22. В велокроссе участвуют 119 спортсменов. Специальное устройство регистрирует прохождение каждым из участников промежуточного финиша, записывая его номер с использованием минимально возможного количества битов, одинакового для каждого спортсмена. Каков информационный объем сообщения, записанного устройством, после того как промежуточный финиш прошли 70 велосипедистов [6]?

Решение.

Найдем количество битов, отводимых на кодирование номера каждого спортсмена. Для этого определим количество битов в числе 119 (максимально возможный номер спортсмена).

Выполним перевод числа 119 в двоичную систему счисления путем делений на 2:

119	2
118	59
1	58
	29
	28
	14
	14
	7
	6
	3
	2
	1

Получим: $119_{10} = 1110111_2$. Следовательно, номер спортсмена кодируется 7 битами ($64 (2^6) < 119 < 128 (2^7)$).

Зная количество битов, отводимых на кодирование одного номера, можно вычислить информационный объем сообщения о номерах 70 велосипедистов:

$$I = 7 \cdot 70 = 490 \text{ бит.}$$

Ответ: 490 бит.

Задача 23. Обычный дорожный светофор без дополнительных секций подает шесть видов сигналов (непрерывные красный, желтый и зеленый; мигающие желтый и зеленый; красный и желтый одновременно). Электронное устройство управления светофором последовательно воспроизводит записанные сигналы. Подряд записано 100 сигналов светофора. Какой объем в байтах составляет информационный объем сообщения [2]?

Решение.

Требуется закодировать 6 сигналов светофора. Допустим, что каждому сигналу соответствует десятичное число в диапазоне от 0 до 5. Число 5 в двоичной системе представляется как 101. Следовательно, для хранения одного сигнала светофора потребуется 3 бита. Поэтому для кодирования 100 сигналов светофора необходимо $100 \cdot 3 = 300$ бит.

Переведем 300 бит в байты: $300 / 8 = 37,5$ байт. Округляя 37,5 в большую сторону, получим, что для кодирования 100 сигналов светофора требуется 38 байтов.

Ответ: 38 байтов.

Задача 24. Метеорологическая станция ведет наблюдение за атмосферным давлением. Результатом одного измерения является целое число, принимающее значение от 720 до 780 мм ртутного столба, которое записывается при помощи минимального количества битов. Станция сделала 80 измерений. Каков информационный объем наблюдений в байтах [4]?

Решение.

Каждое измерение станции — целое число в диапазоне от 720 до 780. Учитывая, что измерение должно записываться с помощью минимального количества битов, сопоставим каждому результату измерения десятичное число в диапазоне от 0 до 60.

Оценим количество битов для кодирования одного измерения: $2^5(32) < 61 < 2^6(64)$.

Получим, что для кодирования одного измерения необходимо 6 битов. Следовательно, информационный объем всех результатов измерений составляет $6 \cdot 80 = 480$ битов = 60 байтов.

Ответ: 60 байтов.

Задача 25. В некоторой стране автомобильный номер состоит из 7 символов. В качестве символов используют 18 различных букв и десятичные цифры в любом порядке.

Каждый такой номер в компьютерной программе записывается минимально возможным и одинаковым целым количеством байтов, при этом используют посимвольное кодирование и все символы кодируются одинаковым и минимально возможным количеством битов. Определите объем памяти, отводимый этой программой для записи 60 номеров [7].

Решение.

Алфавит, используемый для записи номера, состоит из $7 + 18 = 25$ символов. Таким образом, каждый символ номера может принимать одно из 25 значений.

Для кодирования символа номера в двоичной системе необходимо 5 битов, так как $25 < 32$, а $32 = 2^5$.

Таким образом, для хранения одного автомобильного номера требуется $5 \cdot 7 = 35$ битов или 5 байтов ($35 < 5 \cdot 8$).

Следовательно, для записи 60 номеров программа отведет $60 \cdot 5 = 300$ байтов.

Ответ: 300 байтов.

Задача 26. Для кодирования букв А, Б, В, Г решили использовать двухразрядные последовательные двоичные числа (от 00 до 11 соответственно). Что получится, если таким способом закодировать последовательность символов ГБВА и записать результат шестнадцатеричным кодом [4]?

Решение.

Найдем числа, используемые для кодирования указанных букв. Известно, что эти числа идут по порядку начиная с 00. Поэтому букве А ставится в соответствие число 00, Б — 01 ($01 = 00 + 1$), В — 10 ($10 = 01 + 1$), а Г — 11 ($11 = 10 + 1$).

Закодируем последовательность ГБВА: 11 01 10 00.

Переведем полученное число 11011000_2 в шестнадцатеричный код, используя табл. 1.4: $11011000_2 = D8_{16}$.

Ответ: $D8_{16}$.

Задача 27. Для кодирования букв Z, Y, X, W решили использовать одно- и двухразрядные последовательные двоичные числа (от 0 до 11 соответственно). Определите, что получится, если таким способом закодировать последовательность символов WYXZ и записать результат в шестнадцатеричной системе счисления.

Решение.

Найдем числа, используемые для кодирования указанных букв. Известно, что эти числа идут по порядку начиная с 0. Поэтому букве Z ставится в соответствие число 0, Y — 1 ($1 = 0 + 1$), X — 10 ($10 = 1 + 1$), а W — 11 ($11 = 10 + 1$).

Закодируем последовательность WYXZ: 11 1 10 0.

Переведем полученное число 111100_2 в шестнадцатеричный код, используя табл. 1.4: $00111100_2 = 3C_{16}$.

Ответ: $3C_{16}$.

Задача 28. Для 5 букв латинского алфавита заданы их двоичные коды (для некоторых букв — из двух битов, для некоторых — из трех): a — 000, b — 110, c — 01, d — 001, e — 10. Определите, какой набор букв закодирован двоичной строкой 1100000100110 [1].

Решение.

Решение этой задачи основано на декодировании двоичной строки. Одна из сложностей при этом заключается в том, что декодирование может быть неоднозначным, т. е. на определенном шаге декодирования указанной двоичной строке может соответствовать несколько различных наборов букв.

Процесс решения можно изобразить в виде *дерева декодирования* строки, в котором очередное разветвление показывает возможность выбора разных символов на данном этапе декодирования. Анализ задач подобного типа показывает, что обычно при использовании верного направления декодирования (справа налево или слева направо) в них отсутствует неоднозначность выбора очередного символа.

На рис. 1.2 показаны два дерева, на которых изображен процесс анализа строки в различных направлениях. Таким образом, двоичная последовательность 1100000100110 соответствует символьной

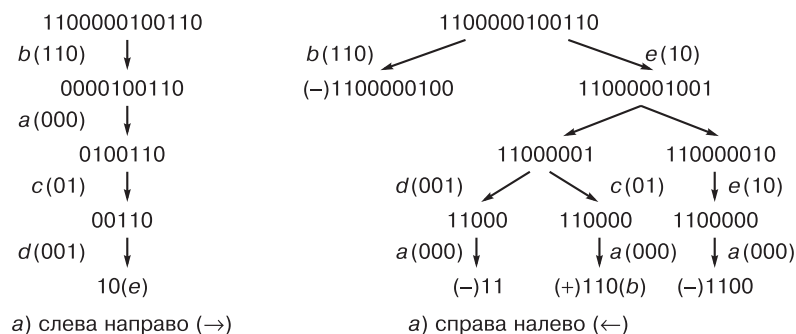


Рис. 1.2. Анализ двоичной строки

строке bacde. (При записи результирующей последовательности символов необходимо учитывать направление анализа.)

Ответ: bacde.

Задача 29. Для 5 букв латинского алфавита заданы их двоичные коды (для некоторых букв — из двух битов, для некоторых — из трех): a — 000, b — 01, c — 100, d — 10, e — 011. Определить, какой набор букв закодирован двоичной строкой 0110100011000 [2].

Решение.

Ниже на рис. 1.3 приведены два дерева, которые отражают процесс декодирования заданной двоичной строки, выполняемый в различных направлениях.

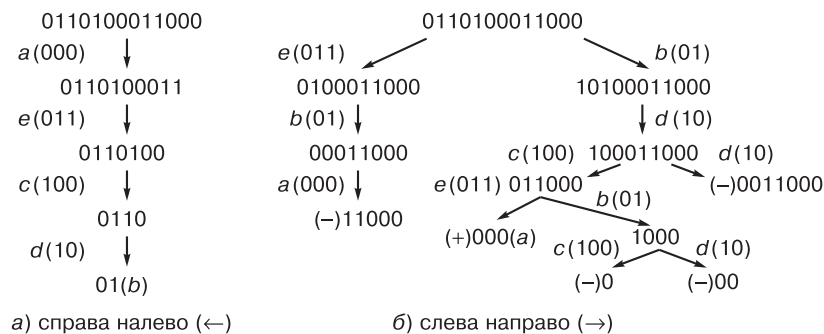


Рис. 1.3. Анализ двоичной строки

Таким образом, двоичная последовательность 0110100011000 соответствует символьной строке bdcea.

Ответ: bdcea.

Задача 30. Для 5 букв русского алфавита заданы их двоичные коды (для некоторых букв — из двух битов, для некоторых — из трех). Эти коды представлены в таблице:

В	К	А	Р	Д
000	11	01	001	10

Определите, будет ли корректно декодировано сообщение 110100001001100111 [3].

Решение.

Декодируем данную двоичную последовательность. Направление декодирования — слева направо (\rightarrow):

11	01	000	01	001	10	01	11
К	А	В	А	Р	Д	А	К

Ответ: последовательность будет корректно декодирована.

Задача 31. Алфавит, в котором записан некоторый текст, включает 8 символов. Ниже приведены вероятности появления каждого символа в тексте: $Z_1 = 0,22$, $Z_2 = 0,20$, $Z_3 = 0,16$, $Z_4 = 0,16$, $Z_5 = 0,10$, $Z_6 = 0,10$, $Z_7 = 0,04$, $Z_8 = 0,02$. Выполните кодирование каждого символа с помощью кода Хаффмана.

Решение.

Запишем таблицу, в которой отразим процесс построения кода.

Буквы	Вероятности	Вспомогательные столбцы						
		1	2	3	4	5	6	7
Z_1	0,22	0,22	0,22	0,26	0,32	0,42	0,58 (1)	1
Z_2	0,20	0,20	0,20	0,22	0,26	0,32 (1)	0,42 (0)	
Z_3	0,16	0,16	0,16	0,20	0,22 (1)	0,26 (0)		
Z_4	0,16	0,16	0,16	0,16 (1)	0,20 (0)			
Z_5	0,10	0,10	0,16 (1)	0,16 (0)				
Z_6	0,10	0,10 (1)	0,10 (0)					
Z_7	0,04 (1)	0,06 (0)						
Z_8	0,02 (0)							

Буквы алфавита записаны в первый столбец в порядке убывания их вероятностей. Далее выполнены следующие шаги:

1) объединим две последние буквы в одну вспомогательную «букву», которой присваивается суммарная вероятность объединяемых букв: $Z_{7,8} = 0,02 + 0,04 = 0,06$;

2) расположим вероятности букв, не участвовавших в объединении, и полученную суммарную вероятность в порядке убывания вероятностей в дополнительном столбце (с номером 2).

Описанный процесс продолжается до тех пор, пока мы не получим единственную вспомогательную букву с вероятностью, равной 1.

Ниже приведено кодовое дерево, которое отображает описанный процесс (рис. 1.4).

Двигаясь по кодовому дереву сверху вниз, запишем для каждой буквы соответствующую ей кодовую комбинацию: $Z_1 = 01$; $Z_2 = 00$; $Z_3 = 111$; $Z_4 = 110$; $Z_5 = 101$; $Z_6 = 1011$; $Z_7 = 10101$; $Z_8 = 10100$.

Ответ: $Z_1 = 01$; $Z_2 = 00$; $Z_3 = 111$; $Z_4 = 110$; $Z_5 = 101$; $Z_6 = 1011$; $Z_7 = 10101$; $Z_8 = 10100$.

Примечание: символы с одинаковыми вероятностями при записи таблицы располагают по алфавиту. Если базовые и вспомогательные символы имеют одинаковые вероятности, то сначала указываются базовые, а затем вспомогательные символы.

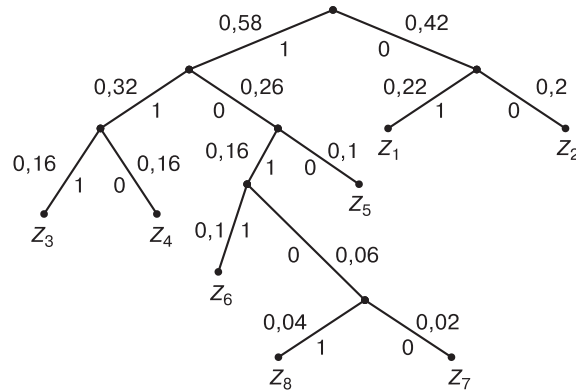


Рис. 1.4. Кодовое дерево

Задача 32. Сгенерируйте семизначный код Хэмминга (7.4) для комбинации 0111.

Решение.

Каждая комбинация семизначного кода Хэмминга включает четыре информационных (отсюда условное обозначение кода: 7.4) и три контрольных символа.

Контрольные символы располагаются в битах, номера которых равны степеням двойки: $1 = 2^0$, $2 = 2^1$, $4 = 2^2$ (это утверждение верно для кода любой длины). Информационные символы располагаются на оставшихся местах. Следовательно, получается следующая запись кода Хэмминга 7.4: $b_1b_2b_3b_4b_5b_6b_7$, где символы с номерами 3, 5, 6 и 7 — информационные, а с номерами 1, 2 и 4 — контрольные.

Заметим, что для некоторого целого числа r существует код Хэмминга (n, m) из $n = 2^r - 1$ символов, содержащий $m = 2^r - r - 1$ символов сообщения.

Определение значений контрольных символов выполняется следующим образом:

1) запишем таблицу, каждый столбец которой будет представлять собой двоичное представление числа 7 (т. е. длины кода):

1	2	3	4	5	6	7
0	0	0	1	1	1	1
0	1	1	0	0	1	1
1	0	1	0	1	0	1

2) запишем уравнения, учитывая только ненулевые биты в каждой строке:

$$b_4 \oplus b_5 \oplus b_6 \oplus b_7 = 0;$$

$$b_2 \oplus b_3 \oplus b_6 \oplus b_7 = 0;$$

$$b_1 \oplus b_3 \oplus b_5 \oplus b_6 = 0;$$

3) значения битов b_4 , b_2 и b_1 вычисляются из первого, второго и третьего уравнений. Для этого нужно выполнить сложение по модулю 2 всех битов, входящих в уравнение, кроме тех, номера которых равны степени двойки, т. е.:

$$b_4 = b_5 \oplus b_6 \oplus b_7;$$

$$b_2 = b_3 \oplus b_6 \oplus b_7;$$

$$b_1 = b_3 \oplus b_5 \oplus b_6.$$

Для исходного информационного сообщения получаются следующие значения контрольных символов:

$$b_4 = 1 \oplus 1 \oplus 1 = 1;$$

$$b_2 = 0 \oplus 1 \oplus 1 = 0;$$

$$b_1 = 0 \oplus 1 \oplus 1 = 0.$$

В результате мы получим следующее кодовое представление исходного сообщения: 0001111.

Ответ: 0001111.

Задача 33. Получена кодовая последовательность 1001111. Известно, что она закодирована кодом Хэмминга и в ней при передаче был искажен один символ. Определите символ, в котором допущена ошибка.

Решение.

Вычислим значения контрольных символов. Если хотя бы один из контрольных символов не будет равен нулю, то это является признаком наличия ошибки.

Определение значений контрольных символов выполняется следующим образом:

1) запишем таблицу, каждый столбец которой будет представлять собой двоичное представление числа 7 (т. е. длины кода):

1	2	3	4	5	6	7
0	0	0	1	1	1	1
0	1	1	0	0	1	1
1	0	1	0	1	0	1

2) запишем уравнения, учитывая только ненулевые биты в каждой строке:

$$b_4 \oplus b_5 \oplus b_6 \oplus b_7 = 0;$$

$$b_2 \oplus b_3 \oplus b_6 \oplus b_7 = 0;$$

$$b_1 \oplus b_3 \oplus b_5 \oplus b_6 = 0;$$

3) значения битов b_4^* , b_2^* и b_1^* вычисляются из первого, второго и третьего уравнений. Для этого сложим все биты, которым соответствуют ненулевые значения в каждой строке таблицы:

$$b_4^* = 1 \oplus 1 \oplus 1 \oplus 1 = 0;$$

$$b_2^* = 0 \oplus 0 \oplus 1 \oplus 1 = 0;$$

$$b_1^* = 1 \oplus 0 \oplus 1 \oplus 1 = 1.$$

Один из контрольных символов получился не равным нулю. Следовательно, ошибка находится в символе с номером $b_4^*b_2^*b_1^* = 001 = 1$. Для исправления ошибки нужно инвертировать первый бит полученного сообщения: 0001111.

Ответ: 1.

1.3.3. Представление чисел в прямом, обратном и дополнительных кодах

В ЭВМ применяются три формы записи (кодирования) целых чисел со знаком: *прямой*, *обратный* и *дополнительный коды*.

Обратный и дополнительный коды обычно применяются для замены используемой в прямом коде операции вычитания операцией сложения, что позволяет упростить конструкцию арифметико-логического устройства ЭВМ.

Положительные числа в прямом, обратном и дополнительном кодах изображаются одинаково — двоичными кодами с цифрой 0 в знаковом разряде.

Пример:

$$1_{10} = \begin{array}{|c|c|c|c|c|c|c|c|} \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ \hline \end{array} \quad 127_{10} = \begin{array}{|c|c|c|c|c|c|c|c|} \hline 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \hline \end{array}$$

\uparrow — знак числа «+» \uparrow — знак числа «+»

Отрицательные числа в прямом, обратном и дополнительном кодах имеют разное изображение. В *прямом* коде числа в знаковый разряд помещается цифра 1, а в разряды цифровой части числа — двоичный код его абсолютной величины.

Пример отрицательных чисел в прямом коде:

$$-1_{10} = \begin{array}{|c|c|c|c|c|c|c|c|} \hline 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ \hline \end{array} \quad \begin{array}{c} \uparrow \\ \text{— знак числа «-»} \end{array} \quad -127_{10} = \begin{array}{|c|c|c|c|c|c|c|c|} \hline 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \hline \end{array} \quad \begin{array}{c} \uparrow \\ \text{— знак числа «-»} \end{array}$$

Можно отметить следующие недостатки представления чисел в прямом коде:

1) в прямом коде возможны два варианта записи числа 0 (например, 00000000 и 10000000 в восьмиразрядном представлении);

2) использование прямого кода для представления отрицательных чисел предполагает или выполнение арифметических операций центральным процессором в прямом коде, или перевод чисел в другое представление (например, в дополнительный код) перед выполнением операций и перевод результатов обратно в прямой код.

Обратный код числа получается инвертированием всех цифр двоичного кода абсолютной величины числа, включая разряд знака. При этом нули заменяются единицами, а единицы — нулями.

Пример:

Число: -1_{10}	Число -127_{10}
Код модуля числа: 0 0000001	Код модуля числа: 0 1111111
Обратный код числа: 1 1111110	Обратный код числа: 1 1111110

Дополнительный код получается из обратного кода последующим прибавлением единицы к его младшему разряду.

Пример:

$$-1_{10} = \begin{array}{|c|c|c|c|c|c|c|c|} \hline 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \hline \end{array} \quad -127_{10} = \begin{array}{|c|c|c|c|c|c|c|c|} \hline 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ \hline \end{array}$$

Обычно при вводе в ЭВМ отрицательные десятичные числа сразу преобразуются в обратный или дополнительный двоичный код и в таком виде хранятся, перемещаются и участвуют в операциях. При выводе же таких чисел выполняется их обратное преобразование в отрицательные десятичные числа.

Арифметические операции над числами в обратном и дополнительном кодах имеют следующие особенности:

1) при сложении чисел в дополнительном коде возникающая единица переноса в знаковом разряде отбрасывается;

2) при сложении чисел в обратном коде возникающая единица переноса в знаковом разряде прибавляется к младшему разряду суммы кодов;

3) если результат арифметических действий является кодом отрицательного числа, то необходимо преобразовать его в прямой код. Обратный код преобразуется в прямой заменой цифр во всех разрядах, кроме знакового, на противоположные. Дополнительный код преобразуется в прямой так же, как и обратный, но с последующим прибавлением единицы к младшему разряду.

Задача 1. Сложить двоичные числа $X = 111$ и $Y = -11$ в обратном и дополнительном кодах.

Решение.

1) Сложим числа по правилам двоичной арифметики:

$$\begin{array}{r} 111 \\ - 11 \\ \hline 100 \end{array}$$

2) Сложим числа, используя коды:

Прямой код:	Обратный код:	Дополнительный код:
$X = 0\ 0000111$	$+ \quad 0\ 0000111$	$+ \quad 0\ 0000111$
$Y = 1\ 0000011$	$\quad 1\ 1111100$	$\quad 1\ 1111101$
	$\hline 1\ 0\ 0000011$	$\hline (1)\ 0\ 0000100$
	$\quad +1$	$\quad 0\ 0000100$
	$\hline 0\ 0000100$	

По правилам сложения в дополнительном коде единица, взятая в круглые скобки, отбрасывается. Результат сложения является кодом положительного числа (в старшем разряде — 0), поэтому:

$$X_{обр} + Y_{обр} = X_{дон} + Y_{дон} = X_{пр} + Y_{пр}.$$

$$\text{Ответ: } X_{обр} + Y_{обр} = 0\ 0000100, X_{дон} + Y_{дон} = 0\ 0000100.$$

Задача 2. Сложить двоичные числа $X = -101$ и $Y = -110$ в обратном и дополнительном кодах.

Решение.

1) Сложим числа по правилам двоичной арифметики:

$$\begin{array}{r} -101 \\ + -110 \\ \hline -1011 \end{array}$$

2) Сложим числа в обратном и дополнительных кодах:

<i>Прямой код</i>	<i>Обратный код:</i>	<i>Дополнительный код:</i>
$X = 1\ 0000101$	$+ \quad 1\ 1111010$	$+ \quad 1\ 1111011$
$Y = 1\ 0000110$	$\quad 1\ 1111001$	$\quad 1\ 1111010$
	$\quad 1\ 1\ 1110011$	$\quad (1)\ 1\ 1110101$
	$\quad \quad \quad +1$	$\quad 1\ 1110101$
	$1\ 1110100$	

По правилам сложения в дополнительном коде единица, взятая в круглые скобки, отбрасывается. Сумма чисел является кодом отрицательного числа (знаковый бит равен 1), следовательно, необходимо перевести результаты в прямой код:

1) из обратного кода:

$$X_{обр} + Y_{обр} = 1\ 1110100 \Rightarrow X_{пр} + Y_{пр} = 1\ 0001011;$$

2) из дополнительного кода :

$$X_{дон} + Y_{дон} = 11110101 \Rightarrow X_{пр} + Y_{пр} = 10001010 + 00000001 = 10001011.$$

Таким образом, $X + Y = -1011$.

Ответ: $X + Y = -1011$.

1.3.4. Модифицированные обратный и дополнительный коды

При переполнении разрядной сетки происходит перенос единицы в знаковый разряд. Это приводит к неверному результату, когда положительное число, получившееся в результате арифметической операции, может восприниматься как отрицательное (так как в знаковом разряде — 1), и наоборот, например:

$$\begin{array}{r}
 + \quad 0\ 1010110\ (X) \\
 \quad 0\ 1101000\ (Y) \\
 \hline
 1\ 0111110
 \end{array}$$

Здесь X и Y — коды положительных чисел, но ЭВМ воспринимает результат их сложения как код отрицательного числа (так как в знаковом разряде стоит 1). Для обнаружения переполнения разрядной сетки вводятся *модифицированные коды*.

В модифицированном обратном и модифицированном дополнительном коде под знак числа отводится не один, а два разряда: 00 соответствует знаку «+», 11 — знаку «-», а любая другая комбинация (01 или 10), получившаяся в знаковых разрядах, является признаком переполнения разрядной сетки.

Сложение чисел в модифицированных кодах ничем не отличается от сложения в обычных обратном и дополнительном кодах.

Выполним сложение в модифицированном обратном коде:

$$\begin{array}{r}
 + \quad 00\ 1010110 \\
 \quad 00\ 1101000 \\
 \hline
 01\ 0111110
 \end{array}$$

Комбинация 01 в знаковых разрядах означает, что произошло переполнение и получившийся результат неверен.

Пример. Даны два числа: $X = 101001$ и $Y = -11010$. Сложить их в модифицированном дополнительном коде.

1) Переведем X и Y в модифицированный дополнительный код:

Обычная запись	Модифицированные	
	обратный код	дополнительный код
$X = +101001$	$X = 00\ 101001$	$X = 00\ 101001$
$Y = -011010$	$Y = 11\ 100101$	$Y = 11\ 100110$

2) Выполним сложение:

$$\begin{array}{r}
 + \quad 00\ 101001 \\
 \quad 11\ 100110 \\
 \hline
 (1)\ 00\ 001111 \\
 \quad 00\ 001111
 \end{array}$$

Переполнения нет (в знаковых разрядах — комбинация 00), поэтому полученный результат правилен ($X + Y = 1111$).

Задача 3. Для хранения целого числа со знаком в ЭВМ используется один байт. Сколько единиц содержит внутреннее представление числа -101 , записанного в прямом коде.

Решение.

Получим двоичное представление числа $101_{10} = 1100101_2$. Так как заданное число отрицательное, то старший бит числа должен быть равен 1. Следовательно, двоичное представление числа -101 будет таким: 11100101_2 . Количество единиц в нем равно 5.

Ответ: 5.

Задача 4. Для хранения целого числа со знаком в ЭВМ используется один байт. Сколько единиц содержит внутреннее представление числа -125 , записанного в прямом коде.

Решение.

Получим двоичное представление числа $125_{10} = 1111101_2$. Заданное число отрицательное, поэтому старший бит числа должен быть равен 1. Следовательно, двоичное представление числа -124 будет таким: 11111101_2 . Количество единиц в нем равно 7.

Ответ: 7.

1.3.5. Представление чисел с фиксированной и плавающей запятой

В ЭВМ применяются две формы представления чисел:

- 1) естественная форма, или форма с фиксированной запятой (точкой);
- 2) нормализованная форма, или форма с плавающей запятой (точкой).

С *фиксированной запятой (точкой)* число изображается в виде последовательности цифр с постоянным положением запятой, отделяющей целую часть от дробной (например: 32,54; 0,0036; $-108,2$).

Эта форма записи наиболее естественна, но имеет небольшой диапазон представления чисел и поэтому не всегда приемлема при вычислениях: если в результате выполнения арифметической операции получится число, выходящее за допустимый диапазон, то происходит переполнение разрядной сетки и дальнейшие вычисления теряют смысл. В современных ЭВМ форма представления чисел с фиксированной запятой используется только для целых чисел (т. е. считается, что в этом случае дробная часть равна нулю).

С *плавающей запятой (точкой)* числа изображаются в виде $X = \pm M \times P^{\pm r}$, где M — *мантисса* числа (правильная дробь; $0,1 \leq M < r$), r — *порядок* числа (целое значение), P — основание системы счисления. Например, приведенные ранее в качестве примера числа с фиксированной запятой можно преобразовать в числа с плавающей запятой так: $32,54 = 0,3254 \times 10^2$; $0,0036 = 0,36 \times 10^{-2}$; $-108,2 = -0,108 \times 10^3$.

Нормализованная форма представления чисел допускает огромный диапазон значений этих чисел и является основной в современных ЭВМ.

В ЭВМ показатель степени принято отделять от мантиссы буквой «Е» (от англ. *exponent*). Например, число $1,528535047 \times 10^{-25}$ в большинстве языков программирования высокого уровня записывается как 1,528535047E-25.

Диапазон чисел, которые можно записать в формате с плавающей запятой, зависит от количества битов, отведенных для представления мантииссы и показателя. В табл. 1.6 приведены возможные варианты записи чисел с плавающей запятой.

Для представления специальных чисел зарезервировано еще два возможных значения показателя: NaN (*Not a Number* — «не число») и $+/-\text{INF}$ (*Infinity* — «бесконечность»), получающихся в результате операций вроде деления на нуль нулевых, положительных и отрицательных чисел.

Таблица 1.6

Варианты записи чисел с плавающей запятой

Точность	Одинарная (single)	Двойная (double)	Расширенная (extended)
Размер (байты)	4	8	10
Число десятичных знаков	7	15	19
Наименьшее значение	$1,4 \times 10^{-45}$	$5,0 \times 10^{-324}$	$1,9 \times 10^{-4951}$
Наибольшее значение	$3,4 \times 10^{+38}$	$1,7 \times 10^{+308}$	$1,1 \times 10^{+4932}$
Поля	S-E-F	S-E-F	S-E-I-F
Размеры полей	1-8-23	1-11-52	1-15-1-63

При этом в табл. 1.6 используются следующие обозначения: S — знак, E — показатель степени, I — целая часть, F — дробная часть.

Задача 1. Значение переменной A представлено в формате с плавающей точкой в шестнадцатеричной системе счисления $A = \text{C34D0000}_{16}$. Тип переменной A — вещественное число с одинарной точностью. Определите десятичное значение числа A .

Решение.

Переведем число A в двоичную систему и запишем его в 32-разрядную ячейку:

$$A_2 = 1100\ 0011\ 0100\ 1101\ 0000000000000000_2.$$

Разряды									Степени															
	7	6	5	4	3	2	1	0	-1	-2	-3	-4	-5	-6	-7	-8	...							
1	1	0	0	0	0	1	1	0	1	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0
S	E (порядок)								F (мантисса)															

Переведем число A_2 в A_{10} по формуле:

$$A_{10} = (-1)^S \cdot 2^{(E-127)} \cdot (1,F). \quad (1.18)$$

Вычислим значения компонентов этой формулы:

1) $S = 1$ (так как число отрицательное);

2) десятичное представление порядка числа: $E = 10000110_2 = 2^7 + 2^2 + 2^1 = 128 + 4 + 2 = 134$;

3) степень двойки: $E - 127 = 7$;

4) выражение $1, F$: F — мантисса числа без незначащих нулей, т. е. $F = 10011010$; единица перед F обозначает целую часть числа; тогда $1, F = 1,10011010 = 2^0 + 2^{-1} + 2^{-4} + 2^{-5} + 2^{-7}$.

Подставим в (1.18) полученные значения компонентов формулы и определим A_{10} :

$$\begin{aligned} A_{10} &= (-1)^1 \cdot 2^{(134 - 127)} \cdot (2^0 + 2^{-1} + 2^{-4} + 2^{-5} + 2^{-7}) = \\ &= -2^7 \cdot (2^0 + 2^{-1} + 2^{-4} + 2^{-5} + 2^{-7}) = \\ &= -(2^7 + 2^6 + 2^3 + 2^2 + 2^0) = -(128 + 64 + 8 + 4 + 1) = -205. \end{aligned}$$

Ответ: -205 .

Задача 2. Значение переменной A представлено в формате с плавающей точкой в шестнадцатеричной системе счисления: $A = 42F20000_{16}$. Тип переменной A — вещественное число с одинарной точностью. Определите десятичное значение числа A .

Решение.

Переведем число A в двоичную систему и запишем его в 32-рядную ячейку:

$$A_2 = 0100\ 0010\ 1111\ 0010\ 0000\ 0000\ 0000\ 0000_2$$

Разряды									Степени																			
	7	6	5	4	3	2	1	0	-1	-2	-3	-4	-5	-6	-7	-8	...											
0	1	0	0	0	0	1	0	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S	E (порядок)								F (мантисса)																			

Переведем число A_2 в A_{10} по формуле:

$$A_{10} = (-1)^S \cdot 2^{(E-127)} \cdot (1, F).$$

Вычислим значения компонент формулы:

1) $S = 0$ (число — положительное);

2) определим десятичное представление порядка числа: $E = 10000101_2 = 2^7 + 2^2 + 2^0 = 128 + 4 + 1 = 133$;

3) вычислим степень двойки: $E - 127 = 6$;

4) запишем выражение $1, F$: F — мантисса числа без незначащих нулей, т. е. $F = 11101000$; единица перед F обозначает целую часть числа; тогда $1, F = 1,11100100 = 2^0 + 2^{-1} + 2^{-2} + 2^{-3} + 2^{-6}$.

$$\begin{aligned}
 A_{10} &= (-1)^0 \cdot 2^{(133-127)} \cdot (2^0 + 2^{-1} + 2^{-2} + 2^{-3} + 2^{-6}) = \\
 &= 2^6 \cdot (2^0 + 2^{-1} + 2^{-2} + 2^{-3} + 2^{-6}) = \\
 &= 2^6 + 2^5 + 2^4 + 2^3 + 2^0 = 64 + 32 + 16 + 8 + 1 = 121.
 \end{aligned}$$

Ответ: 121.

1.3.6. Арифметические операции над числами

Сложение

Сложение чисел в q -ичной системе счисления основано на поразрядном сложении чисел с переносом из младших разрядов в старшие.

При сложении цифр можно использовать следующий способ. Предположим, что необходимо сложить две цифры в шестнадцатеричной системе счисления — F_{16} и D_{16} . Тогда нужно:

1) сложить цифры в десятичной системе счисления: $15(F) + 13(D) = 28$;

2) записать результат сложения в развернутой форме записи числа с учетом того, что результат должен быть представлен в шестнадцатеричной системе счисления:

$$28 = 1 \cdot 16^1 + 12 \cdot 16^0 = 1C_{16}.$$

Следовательно, $F_{16} + D_{16} = 1C_{16}$.

Задача 1. Сложить числа 205_8 и 374_8 .

Решение.

Рассмотрим процесс сложения двух чисел в восьмеричной системе счисления:

<i>Перенос</i>	110
	374
+	205
<i>Результат</i>	601

1) « $4 + 5$ »: сложим цифры 4 и 5 в десятичной системе счисления — $4 + 5 = 9$. Число 9 можно представить как $9 = 1 \cdot 8^1 + 1 \cdot 8^0$, т. е. $9_{10} = 11_8$. Тогда 1 записываем в ответ, а 1 переносим в старший разряд;

2) « $1 + 7 + 0$ »: складываем 1 и 7 получаем $8_{10} = 10_8$. В ответ записываем 0 и переносим в старший разряд 1;

3) « $1 + 3 + 2$ »: получаем 6 в восьмеричной системе счисления.

Ответ: $205_8 + 374_8 = 601_8$.

Вычитание

Вычитание чисел в q -ичной системе счисления выполняется поразрядно, причем в случае необходимости выполняется заем 1 из старшего разряда. При этом занимаемая единица старшего разряда равна q единицам младшего разряда (например, в случае двоичной системы счисления 1 старшего разряда равна 2 единицам младшего разряда).

Задача 2. Вычислить разность чисел 250_8 и 55_8 .

Решение.

Рассмотрим процесс вычитания:

Заем	$1 \rightarrow$ ($2 = 1 + 1$)	$1 \rightarrow$ ($5 = 1 + 4$)	8
—	2	5	0
	0	5	5
Результат	1	7	3

1) «0 – 5»: поскольку из 0 вычесть 5 нельзя, занимаем в старшем разряде 1 (т. е. занимаем у 5). В младший разряд эта единица переносится как 8, так как одна единица старшего разряда в восьмеричной системе счисления равна восьми единицам младшего разряда. Соответственно, в старшем разряде от 5 остается 4 единицы;

2) «4 – 5»: из 4 вычесть 5 нельзя, поэтому занимаем единицу в старшем разряде и получаем: $8 + 4 - 5 = 7$. В старшем разряде остается 1;

3) «1 – 0»: из единицы вычитаем 0 и получаем 1.

Ответ: $250_8 - 55_8 = 173_8$.

Умножение

Умножение чисел, находящихся в q -ичной системе счисления, выполняется поразрядно с переносами в старшие разряды. При перемножении цифр числа можно использовать следующий способ. Предположим, что необходимо перемножить две цифры в шестнадцатеричной системе счисления — F_{16} и D_{16} . Тогда нужно:

1) перемножить цифры в десятичной системе счисления: $15 \cdot 13 (F \cdot D) = 195$;

2) записать результат перемножения (число 195) в развернутой форме с учетом того, что результат должен быть представлен

в шестнадцатеричной системе счисления, и привести полученные цифры в соответствие с алфавитом шестнадцатеричной системы: $195 = 12 \cdot 16 + 3$ (т. е. $12 \cdot 16^1 + 3 \cdot 16^0$). Следовательно, $F_{16} \cdot D_{16} = C3_{16}$.

Задача 3. Умножить числа 205_8 и 374_8 .

Решение.

Рассмотрим выполняемые при решении этой задачи действия.

$$\begin{array}{r} \times \quad 25 \\ \hline 37 \\ + 223 \\ \hline 77 \\ \hline 1213 \end{array}$$

1) « $7 \cdot 5$ »: перемножим 7 и 5 в десятичной системе счисления и получим число 35. Запишем 35 в восьмеричной системе счисления: $35_{10} = 4 \cdot 8^1 + 3 \cdot 8^0$, т. е. $35_{10} = 43_8$. Следовательно, в качестве промежуточного результата записываем 3 под цифрой 7, а 4 переносится в старший разряд;

2) « $7 \cdot 2$ »: перемножим 7 и 2 в десятичной системе и получим 14. В восьмеричной системе 14 представляется как $14_{10} = 1 \cdot 8^1 + 6 \cdot 8^0 = 16_8$. Так как в предыдущем действии в старший разряд был выполнен перенос 4, сложим 16_8 и 4_8 и запишем результат под цифрой 3: $16_8 + 4_8 = 22_8$. В результате мы получаем результат умножения 7 на 25 — число 223;

3) « $3 \cdot 5$ »: перемножим 3 и 5 в десятичной системе счисления и получим 15. В восьмеричной системе 15 представляется как $15_{10} = 1 \cdot 8^1 + 7 \cdot 8^0 = 17_8$. Следовательно, в качестве промежуточного результата запишем 7 под цифрой 3, а 1 переносится в старший разряд;

4) « $3 \cdot 2$ »: перемножим 3 и 2 в восьмеричной системе счисления и получим 6. Кроме того, в предыдущем действии был выполнен перенос 1 в старший разряд. Следовательно, в промежуточном результате мы запишем сумму 1 и 6, т. е. число 7. В результате получаем результат умножения 30 на 25 — число 770;

5) « $223 + 770$ »: сложим поразрядно числа 223 и 770 и получим число 1213.

Ответ: $205_8 \cdot 374_8 = 1213_8$.

Деление

Деление чисел в любой системе счисления основано на последовательном применении операций умножения и вычитания.

Задача 4. Вычислить в восьмеричной системе счисления результат деления $1224,62_8$ на 12_8 .

Решение.

Рассмотрим процесс деления.

$$\begin{array}{r}
 \begin{array}{r} 1224,62 \\ 12 \end{array} \overline{) 12} \quad \begin{array}{r} 12 \\ 0102,05 \end{array} \\
 \underline{24} \\
 62 \\
 \underline{62} \\
 0
 \end{array}$$

1) сформируем неполное делимое — запишем в него первую цифру делимого — 1. Так как 1 меньше 12, то запишем в частное 0, а к неполному делимому 1 допишем следующую цифру делимого, т. е. 2;

2) полученное неполное делимое $12 \geq 12$. Тогда найдем такой множитель n для делителя, что $(12 - 12 \cdot n) \leq 12$. Таким числом является 1, поэтому запишем 1 в частное;

3) вычтем результат перемножения делителя 12 на 1 из неполного делимого 12 и получим 0. (Весь процесс записывается с учетом разрядов.);

4) сформируем новое неполное делимое: запишем в него очередную цифру делимого — 2. Так как 2 меньше 12, то в частное запишем 0, а к неполному делимому 2 допишем следующую цифру делимого, т. е. 4;

5) полученное неполное делимое $24 \geq 12$. Найдем такой множитель n для делителя, что $(24 - 12 \cdot n) \leq 12$. Таким числом является 2, поэтому запишем 2 в частное. Вычтем результат перемножения делителя 12 на 2 из неполного делимого 24 и получим 0;

6) в целой части делимого цифры закончились, поэтому поставим в частном запятую. Вновь сформируем неполное делимое: запишем в него первую цифру делимого, находящуюся после запятой, — 6. Так как 6 меньше 12, то в частное запишем 0, а к 6 допишем следующую цифру делимого, т. е. 2;

7) полученное неполное делимое $62 \geq 12$. Найдем такой множитель n для делителя, что $(62 - 12 \cdot n) \leq 12$. Таким числом является 5, поэто-

му запишем 5 в частное. Вычтем результат перемножения делителя 12 на 5 из неполного делимого 62 и получим 0.

Деление завершено.

Ответ: $1224,62_8 / 12_8 = 102,5$.

Внимание! При выполнении всех действий необходимо учитывать, что они производятся в восьмеричной системе счисления.

Задача 5. Вычислите значение суммы $10_2 + 10_8 + 10_{16}$ в двоичной системе счисления [1].

Решение.

Переведем каждое слагаемое в двоичную систему счисления, используя табл. 1.4 и 1.5:

$$10_8 = 001\,000_2 = 1000_2, \quad 10_{16} = 0001\,0000_2 = 10000_2.$$

Выполним сложение полученных чисел:

$$10_2 + 10_8 + 10_{16} = 10_2 + 1000_2 + 10000_2 = 11010_2.$$

<i>Перенос</i>	00000
	10000
+	01000
	00100
<i>Результат</i>	11010

Ответ: 11010_2 .

Задача 6. Вычислите сумму чисел x и y , если $x = 1010101_2$, $y = 1010011_2$.

Решение.

Сложим числа x и y :

<i>Перенос</i>	10101110
	01010101
+	01010011
<i>Результат</i>	10101000

Получаем: $x + y = 1010101_2 + 1010011_2 = 10101000_2$.

Ответ: 10101000_2 .

Задача 7. Вычислите сумму чисел x и y при $x = 1D_{16}$, $y = 72_8$. Результат представьте в двоичной системе счисления [3].

Решение.

Переведем заданные числа в двоичную систему счисления:

$$1) 1D_{16} = 0001\,1101_2 = 011101_2;$$

$$2) 72_8 = 111\,010_2 = 111010_2.$$

Сложим эти числа:

Перенос	1110000
+	0011101
	0111010
Результат	1010111

Получаем: $1D_{16} + 72_8 = 1010111_2$.

Ответ: 1010111_2 .

1.4. Кодирование текстовой информации

Кодирование текстовой информации в ЭВМ основано на использовании *таблицы кодировки*, в которой устанавливается соответствие между символом (буквой, знаком препинания, цифрой, графическим символом) и его уникальным десятичным кодом.

Базовым стандартом кодирования текстовой информации является *стандарт ASCII (American Standard Code for Information Interchange)*, разработанный в США в Национальном институте ANSI (American National Standards Institute). В стандарте ASCII на кодирование одного символа отводится 1 байт, поэтому, используя кодировку ASCII, можно закодировать 256 различных символов.

В стандарте ASCII описываются две таблицы — базовая и расширенная. Базовая таблица закрепляет значения кодов от 0 до 127, а расширенная относится к символам с номерами от 128 до 255. При этом первые 33 кода (с 0 до 32) соответствуют не символам, а операциям (перевод строки, ввод пробела и т. д.). Коды с 33 по 127 являются международными и соответствуют символам латинского алфавита, цифрам, знакам арифметических операций и знакам препинания. Коды же с 128 по 255 являются национальными, т. е. в национальных кодировках одному и тому же коду соответствуют различные символы.

Кодирование символов русского алфавита может осуществляться на основе нескольких кодовых таблиц (КОИ-8R, CP-1251, CP-866, Mac, ISO-8859-5), которые несовместимы друг с другом.

Кодировка CP-866 основана на реализации таблицы ASCII, разработанной фирмой IBM. Эта кодировка в верхней половине расширенной кодовой таблицы содержит псевдографические символы, а специфические европейские символы в верхней половине кодовой таблицы заменены на кириллицу. Данная кодировка создана в ВЦ АН СССР, для которого впервые в СССР была закуплена партия IBM PC.

Кодировка КОИ-8R разработана при адаптации операционной системы UNIX к русскому языку. В ней символы русской кириллицы размещены в верхней части расширенной ASCII таблицы так, что позиции кириллических символов соответствуют их фонетическим аналогам в английском алфавите в нижней части таблицы.

Семейство кодировок ISO-8859-X определено международной организацией по стандартизации (ISO). Это семейство представляет собой совокупность 8-битных кодировок, где младшая половина кодовой таблицы (символы с кодами 0–127) соответствует таблице ASCII, а в верхней половине определены символы для различных языков. Например, кодовая страница для кириллицы определена в стандарте ISO-8859-5.

Кодировка CP1251 — стандартная 8-битная кодировка для всех русских версий Microsoft Windows, которая создана на базе кодировок, использовавшихся в ранних русификаторах Windows.

Кодировка UNICODE была разработана для создания единой кодировки символов всех современных и многих древних письменных языков. Каждый символ в этом стандарте кодируется 16 битами, что позволяет охватить гораздо большее количество символов, чем принятые ранее 7- и 8-битовые кодировки. В кодировке UNICODE с каждым символом связан уникальный код и определены характеристики этого символа, например:

- тип символа (прописная буква, строчная буква, цифра, знак препинания и т. д.);
- атрибуты символа (отображение слева направо или справа налево, пробел, разрыв строки и т. д.);
- соответствующая прописная или строчная буква (для строчных и прописных букв);
- соответствующее числовое значение (для цифровых символов).

Таблица 1.7

Характеристики кодировок символов

Название	Байт/символ
CP-866	1
КОИ-8R	1
ISO	1
CP-1251	1
UNICODE	2

Объем информации, содержащийся в тексте из K символов, равен:

$$I_t = K \cdot I, \quad (1.19)$$

где I_t — объем информации в тексте, а I — информационный вес одного символа текста (количество байтов, отводимых на хранение одного символа).

Задача 1. Считая, что каждый символ кодируется одним байтом, оцените информационный объем в битах следующего предложения: «Мой дядя самых честных правил. Когда не в шутку занемог, Он уважать себя заставил И лучше выдумать не мог». [1].

Решение.

Для оценки информационного объема предложения необходимо сосчитать количество символов (букв, а также знаков препинания — кавычек, точек, запятых — и пробелов), входящих в предложение: букв — 85, пробелов — 18, знаков препинания — 5. Получаем, что предложение состоит из 108 символов. Учитывая, что каждый символ кодируется одним байтом, можно записать: $108 \cdot 8 = 864$ бита.

Ответ: 864 бита.

Задача 2. Автоматическое устройство осуществило перекодировку информационного сообщения на русском языке, первоначально записанного в 16-битном коде Unicode, в 8-битную кодировку КОИ-8. При этом информационное сообщение уменьшилось на 480 бит. Какова длина сообщения в символах [5]?

Решение.

Пусть x — количество символов в информационном сообщении. Тогда информационный объем сообщения в кодировке UNICODE составляет $16 \cdot x$ битов, а в кодировке КОИ-8 — $8 \cdot x$ битов. Исходя из условия задачи, запишем следующее уравнение:

$$16 \cdot x - 8 \cdot x = 480 \Rightarrow 8 \cdot x = 480 \Rightarrow x = 480 / 8 \Rightarrow x = 60 \text{ (символов)}.$$

Ответ: 60 символов.

Задача 3. В таблице ниже 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

Определите шестнадцатеричный код символа «q» [7].

Решение.

Заглавный и строчный символы находятся в таблице ASCII на одном расстоянии:

1) $K_a - K_A = 97 - 65 = 32$, где K_a — код символа «a», K_A — код символа «A»;

2) $K_b - K_B = 98 - 66 = 32$, где K_b — код символа «b», K_B — код символа «B».

Следовательно, зная код символа «Q», вычислить код символа «q» можно следующим образом:

$$K_q - K_Q = 32 \Rightarrow K_q = K_Q + 32 = 81 + 32 = 113.$$

Переведя 113 в шестнадцатеричную систему, получим: $113_{10} = 71_{16}$.

Ответ: 71_{16} .

1.5. Кодирование графической информации

Графическая информация может быть представлена в аналоговой или дискретной форме. Примером аналогового представления графической информации является, например, картина на холсте, а примером дискретного представления — изображение на экране монитора, состоящее из отдельных точек — *пикселей* (*pixel* — *PIcture ELement*) разного цвета.

Получение цифрового представления изображения основано на выполнении *пространственной дискретизации* аналогового изображения (осуществлении *аналога-цифрового преобразования*). Данный процесс заключается в разбиении непрерывного (аналогового) изображения на отдельные мелкие фрагменты, после чего цвет каждого фрагмента (а точнее — код цвета, например в *цветовой системе RGB*) записывается в ячейку таблицы с координатами, соответствующими координатам фрагмента исходного изображения.

Одним из устройств, которое выполняет дискретизацию изображения, является *сканер*.

Сканер — это устройство для ввода в ЭВМ графической информации. Сканер осуществляет аналого-цифровое преобразование. К основным параметрам, определяющим результат работы сканера, относятся:

1) *оптическое разрешение* измеряется в точках на дюйм (*dots per inch* — *dpi*). Обычно указывается два значения, например 600×1200 dpi, где горизонтальное разрешение (первая цифра) опре-

деляется *CCD-матрицей** сканера, а вертикальное (вторая цифра) определяется количеством шагов двигателя на дюйм;

2) *глубина цвета* определяется качеством CCD-матрицы и разрядностью АЦП. Измеряется количеством оттенков, которые устройство способно распознать (например, 24 бита соответствуют 16 777 216 оттенкам). В настоящее время сканеры выпускают с глубиной цвета 24, 30 и 36 бит.

Цифровое изображение обычно описывается следующими параметрами:

1) *глубина цвета* — количество битов, используемых для представления цвета при кодировании одного пикселя изображения:

$$I = \log_2 N, \quad (1.20)$$

где N — количество цветов в изображении, а I — глубина цвета;

2) *цветовой диапазон* — максимальное количество цветов в изображении:

$$N = 2^I; \quad (1.21)$$

3) *размер изображения* — количество пикселей по вертикали (w) и по горизонтали (h);

4) *объем памяти, занимаемой изображением*:

$$I_I = I \cdot h \cdot w, \quad (1.22)$$

где I_I — объем памяти, занимаемой изображением, а I — глубина цвета.

Описание цветов в ЭВМ основано на использовании цветовых моделей и соответствующих им способов кодирования цвета.

Модель RGB. Название этой цветовой модели происходит от названий трех используемых в ней базовых цветов — *Red* (красный), *Green* (зеленый) и *Blue* (синий). Данная модель является *аддитивной*, т. е. требуемый произвольный цвет получается при *сложении* трех базовых цветов. Яркость каждого базового цвета может при этом принимать значения от 0 до 255 (256 значений); таким образом, данная модель позволяет кодировать $256 \cdot 256 \cdot 256 = 2^8 \cdot 2^8 \cdot 2^8 = 2^{24}$ цветов (см. табл. 1.8). Если значения яркостей всех базовых цветов равны, то образуемый цвет представляет собой один из оттенков серого.

* *CCD-матрица (Charge-Coupled Device)* — специализированная аналоговая интегральная микросхема, состоящая из светочувствительных фотодиодов, выполненная на основе кремния и использующая технологию ПЗС — приборов с зарядовой связью.

Таблица 1.8

Кодирование цветов в системе RGB

Значение базового цвета			Образуемый цвет
R	G	B	
255	0	0	красный
0	255	0	зеленый
0	0	255	синий
255	255	255	белый
0	0	0	черный
255	255	0	желтый
0	255	255	голубой
255	0	255	пурпурный

Модель СМΥК. Данная цветовая модель — основная в полиграфии. Пурпурный (*Magenta*), голубой (*Cyan*) и желтый (*Yellow*) цвета составляют *полиграфическую триаду*. При печати этими красками большая часть видимого цветового спектра может быть воспроизведена на бумаге.

В модели СМΥК основные цвета образуются путем *вычитания* из белого цвета цветов модели RGB: голубой (белый минус красный: $C = W - R$), пурпурный (белый минус зеленый: $M = W - G$), желтый (белый минус синий: $Y = W - B$). Для улучшения качества получаемого изображения в число основных полиграфических красок (и в данную цветовую модель) добавлен черный цвет. Название черной компоненты в аббревиатуре «СМΥК» сокращается до буквы К, поскольку эта краска является главной, ключевой (*Key*) в процессе цветной печати (а также чтобы не путать букву *B* от *Black* с буквой названия синего цвета — *Blue*). Интенсивность каждого цветового компонента может быть выражена в процентах или в градациях от 0 до 255, при этом для кодирования цвета одного пикселя требуется 32 бита (4 байта).

Модели RGB и СМΥК связаны между собой, но изменение цветовой модели изображения обычно невозможно без искажений цветов в изображении. Существуют также и другие цветовые модели, например LAB, HSV (HSB), YUV и т. д.

Рассмотрим цветовые схемы, используемые при создании изображений. Одной из простейших является схема, в которой каждая точка изображения может иметь только два цвета — «черный» или «белый», т. е. для хранения цвета пикселя необходим 1 бит.

Изображения, состоящие из оттенков серого цвета, называют *полутоновыми*. Каждый пиксель полутонового изображения может иметь один из 256 оттенков серого — от черного (0) до белого (255). Для хранения цвета каждого пикселя при этом требуется 1 байт. Полутоновые изображения широко используются для хранения черно-белых (в традиционном, фотографическом смысле) фотографий и в тех случаях, когда без цветов можно обойтись. Глубина цвета полутоновых изображений равна 8.

При кодировании цвета в *полноцветных* изображениях с использованием модели RGB обычно используются цветовые схемы, указанные в таблице ниже:

Количество бит на кодирование цвета пикселя	Описание схемы
8	R — 3 бита, G — 3 бита, B — 2 бита ($2^3 \times 2^3 \times 2^2$ цветов). Человеческий глаз менее чувствителен к синей составляющей, чем к красной и зеленой, поэтому для представления синей составляющей используется на один бит меньше
12	R — 4 бита, G — 4 бита, B — 4 бита ($2^4 \times 2^4 \times 2^4$ цветов)
HighColor	а) 15-битный цвет: R — 5 бит, G — 5 бит, B — 5 бит ($2^5 \times 2^5 \times 2^5$ цветов) б) 16-битный цвет: R — 5 бит, G — 6 бит, B — 5 бит ($2^5 \times 2^6 \times 2^5$ цветов)
TrueColor (24)	R — 8 бит, G — 8 бит, B — 8 бит ($2^8 \times 2^8 \times 2^8$ цветов)
32	32-битный цвет является 24-битным (True Color) с дополнительным 8-битным каналом, который либо заполнен нулями, либо представляет собой <i>альфа-канал</i> , определяющий прозрачность изображения в определенных пикселях ($2^8 \times 2^8 \times 2^8 \times 2^8$ цветов)

В зависимости от выбранного способа кодирования графической информации различают векторные и растровые изображения.

Векторное изображение содержит объекты, определяемые математическими уравнениями, которые содержат информацию о размере, форме, цвете, границе и местоположении каждого объекта. Достоинства векторных изображений — малый объем файлов и отсутствие потерь качества при масштабировании.

Возможные форматы файлов векторных изображений:

- 1) *WMF* — универсальный формат векторных графических файлов для Windows-приложений; может иметь глубину цвета до 24 бит и хранить наряду с векторными изображениями также растровые фрагменты;
- 2) *SVG* — формат хранения векторных изображений, предназначенный для описания двумерной векторной и смешанной векторно/растровой графики.

Растровое изображение представляет собой совокупность (матрицу) пикселей, каждый из которых имеет определенный цвет в заданной цветовой модели. Растровая графика наиболее распространена там, где требуется создание детализированных реалистичных изображений со множеством оттенков.

Особенности растровых изображений:

- 1) большой объем занимаемой памяти по сравнению с векторными изображениями, так как требуется хранение данных о каждом пикселе изображения;
- 2) масштабирование растрового изображения ухудшает его качество.

Возможные форматы файлов растровых изображений:

- 1) *JPEG (Joint Photographic Experts Group)* — разработан для хранения растровых изображений со сжатием;
- 2) *BMP (Windows Bitmap)* — формат хранения растровых изображений без сжатия; глубина цвета — от 1 до 48 бит на пиксель; максимальные размеры изображения — 65535×65535 пикселей;
- 2) *TIFF (Tagged Image Format)* — поддерживает большой диапазон изменения глубины цвета (1, 4, 8, 24, 48 бит), разные цветовые пространства и настройки сжатия (как с потерями, так и без);
- 4) *RAW* — хранит информацию, получаемую непосредственно с матрицы цифрового фотоаппарата или аналогичного устройства без применения к ней каких-либо преобразований.

Задача 1. Сколько секунд потребуется модему, передающему информацию со скоростью 16 000 бит/с, чтобы передать 8-цветное растровое изображение размером 800×600 пикселей, при условии что в одном байте закодировано максимально возможное целое число пикселей?

Решение.

Зная количество цветов изображения, вычислим количество битов, отводимых на кодирование цвета каждого пикселя, используя формулу (1.20):

$$I = \log_2 8 = 3 \text{ (бита)}.$$

Далее, используя формулу (1.22), вычислим объем изображения в битах:

$$I_I = 800 \cdot 600 \cdot 3 \text{ (бит)}.$$

На основе формулы (1.15) вычислим время передачи файла:

$$T = 800 \cdot 600 \cdot 3 / 16\,000 = 8 \cdot 18 \cdot 1000 / 16\,000 = 144 / 16 = 9 \text{ (секунд)}.$$

Ответ: 9 секунд.

Задача 2. Для хранения растрового изображения размером 16×32 пикселя отвели 1 килобайт памяти. Каково максимально возможное число цветов в палитре изображения?

Решение.

Используя формулу (1.22) и переводя килобайты в байты, можно записать:

$$1024 = I \cdot 16 \cdot 32 \Rightarrow I = 1024 / (16 \cdot 32) = 2^{10} / (2^4 \cdot 2^5) \text{ (байт)}.$$

Переведем байты в биты и запишем:

$$I = 2^{10} \cdot 2^3 / 2^{4+5} = 2^{13} / 2^9 = 2^4 = 16 \text{ (бит)}.$$

Таким образом, на кодирование цвета одного пикселя отводится 16 бит. Оценим количество цветов в палитре изображения на основе формулы (1.21):

$$N = 2^{16} = 65\,536 \text{ (цветов)}.$$

Ответ: 65 536 цветов.

Задача 3. Цвет пикселя, формируемого принтером, определяется тремя составляющими: голубой, пурпурной и желтой. Под каждую составляющую одного пикселя отвели по 3 бита. В какое количество цветов можно раскрасить пиксель?

Решение.

Учитывая, что на каждую из трех цветовых составляющих отводится 3 бита, цвет пикселя кодируется 9 битами. Используя формулу (1.21), вычислим количество цветов, в которые можно раскрасить пиксель: $N = 2^9$. Используя табл. 1.2, вычислим $N = 2^9 = 512$ цветов.

Ответ: 512 цветов.

Задача 4. Цвет пикселя изображения определяется тремя составляющими: зеленой, синей и красной. Под красную и синюю составляющие пикселя отвели по 10 битов. Сколько битов отвели для хранения зеленой составляющей пикселя, если растровое изображение размером 4×4 пикселя занимает 128 байт памяти?

Решение.

Вычислим количество битов для кодирования цвета пикселя, используя формулу (1.20):

$$128 = I \cdot 4 \cdot 4 \Rightarrow I = 128 / 16 = 2^7 / 2^4 = 8 \text{ (байт)} = 64 \text{ (бит)}.$$

На красную и синюю составляющие отвели по 5 битов, следовательно, на зеленую составляющую отводится: $64 - 10 - 10 = 44$ бита.

Ответ: 44 бита.

Задача 5. Укажите минимальный объем памяти (в килобайтах), достаточный для хранения любого растрового изображения размером 32×32 пикселя, если известно, что в изображении используется палитра из 256 цветов. Саму палитру хранить не нужно.

Решение.

Вычислим количество битов, отводимых на кодирование цвета одного пикселя, используя формулу (1.20):

$$I = \log_2 256 = 8 \text{ бит}.$$

На основе формулы (1.22) рассчитаем объем памяти для хранения изображения:

$$I_I = 8 \cdot 32 \cdot 32 = 2^3 \cdot 2^5 \cdot 2^5 = 2^{3+5+5} = 2^{13} \text{ (бит)}.$$

Используя табл. 1.1, переведем 2^{13} бит в килобайты (1 килобайт = 2^{13} бит): $2^{13} / 2^{13} = 1$ (кб).

Ответ: 1 кб.

Задача 6. Для кодирования цвета фона web-страницы используется атрибут `bgcolor="#XXXXXX"`, где в кавычках задаются шестнадцатеричные значения интенсивности цветовых компонент в 24-битной RGB-модели. Какой цвет будет у страницы, заданной тегом `<body bgcolor="#00FF00">` [7]?

Решение.

Кодирование каждой цветовой компоненты выполняется шестнадцатеричным кодом. Следовательно, используя табл. 1.8 и зная, что $\text{FF}_{16} = 255_{10}$, можно получить следующие значения базовых цветов: $R = 0$, $G = \text{FF} (255)$, $B = 0$. Таким образом, у страницы, заданной тегом `<body bgcolor="#00FF00">`, будет зеленый цвет фона.

Ответ: зеленый цвет.

Задача 7. Для кодирования цвета фона web-страницы используется атрибут `bgcolor="#XXXXXX"`, где в кавычках задаются шестнадцатеричные значения интенсивности цветовых компонент в 24-битной RGB-модели. Какой цвет будет у страницы, заданной тегом `<body bgcolor="#909090">`?

Решение.

Из условия задачи получим значения базовых цветов: $R = 90$, $G = 90$, $B = 90$. Как известно, если используется RGB модель и значения яркостей всех базовых цветов одинаковы, то цвет страницы, задаваемой тегом `<body bgcolor="#909090">`, будет серым.

Ответ: серый цвет.

Задача 8. Цветной сканер имеет разрешение 256×128 точек/дюйм. Объем памяти, занимаемой отсканированным изображением размером 2×2 дюйма, составляет 2 Мб. Определите глубину представления цвета в битах.

Решение.

Определим размер отсканированного изображения в точках: $h = 256 \cdot 2 = 2^9$ точек, $w = 128 \cdot 2 = 2^8$ точек.

Используя формулу (1.22), запишем выражение для определения глубины цвета в битах:

$$I = I_t / (h \cdot w).$$

Получаем:

$$\begin{aligned} I &= 2 \cdot 1024 \cdot 1024 \cdot 8 / (2^9 \cdot 2^8) = 2 \cdot 2^{10} \cdot 2^{10} \cdot 8 / (2^9 \cdot 2^8) = \\ &= 2 \cdot 8 \cdot 2^3 = 128 \text{ битов.} \end{aligned}$$

Ответ: 128 битов.

1.6. Кодирование звуковой информации

Слуховая система человека способна воспринимать упругие волны, имеющие частоту в пределах от 16 Гц до 20 кГц. Упругие волны в любой среде, частоты которых лежат в указанных пределах, называют звуковыми. Регистрация звуковых волн (звука) выполняется с помощью *микрофона*, который выполняет преобразование звуковой волны в электрический сигнал. Поскольку этот электрический сигнал на выходе микрофона непрерывен, для его ввода в ЭВМ нужно преобразовать его в цифровую форму, т. е. выполнить дискретизацию сигнала.

Дискретизация сигнала во времени — это преобразование непрерывного аналогового сигнала в последовательность его значений в дискретные моменты времени. Эти значения называются *отсчетами*, или *выборками*. Данную функцию выполняет *аналого-цифровой преобразователь (АЦП)*.

Аналого-цифровой преобразователь осуществляет:

1) дискретизацию сигнала по времени, т. е. измерение уровня интенсивности звука в определенные фиксированные моменты времени. Частоту, характеризующую периодичность измерения звукового сигнала, принято называть *частотой дискретизации*. Частота дискретизации выбирается на основе *теоремы Котельникова* и должна быть как минимум в 2 раза больше максимальной частоты спектра сигнала. Вместе с тем считается, что человек не слышит звук с частотой более 20 000 Гц (20 кГц), поэтому для высококачественного воспроизведения звука верхнюю границу обычно принимают равной 22 кГц. Следовательно, частота дискретизации сигнала должна быть не меньше 44 кГц;

2) дискретизацию амплитуды звукового сигнала, т. е. представление диапазона интенсивности звука с помощью набора уровней (например, 256 или 65536); текущий уровень измеряемого сигнала округляется до ближайшего из этих уровней.

Параметры дискретного звукового сигнала:

- 1) длительность сигнала t (измеряется в секундах);
- 2) глубина кодирования звука I (измеряется в битах) — количество битов, отводимых для хранения одного отсчета дискретного сигнала. Обычно используют 8-, 16- или 20-битное представление значений амплитуды;
- 3) частота дискретизации F (измеряется в герцах) — количество измерений амплитуды аналогового сигнала в секунду;
- 4) количество звуковых каналов записи N ($N = 1$ — монозапись, $N = 2$ — стереозапись);
- 5) размер файла: I_s (измеряется в байтах):

$$I_s = F \cdot I \cdot t \cdot N. \quad (1.23)$$

На рис. 1.5 показан пример дискретизации аналогового сигнала $f(x)$ ($f(x) \in [0; 1]$) длительностью в 1 секунду. Частота дискретизации — 10 герц, т. е. из аналогового сигнала длительностью в 1 секунду формируется дискретный сигнал, состоящий из 10 отсчетов. Глубина кодирования — 4 бита, т. е. всего существует 16 возмож-

ных значений амплитуды: $0/16, 1/16, 2/16, \dots, 14/16, 15/16$. Цифрами на рисунке надписаны номера отсчетов дискретного сигнала.

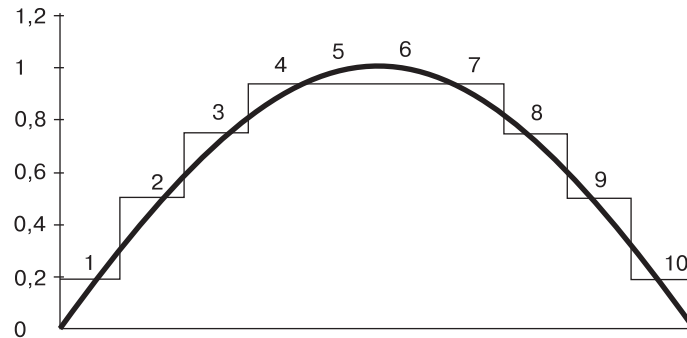


Рис. 1.5. Пример дискретизации сигнала

Задача 1. Определить объем памяти для хранения цифрового аудиофайла, содержащего стереозапись, время звучания которого составляет одну секунду при частоте дискретизации 48 кГц и разрешении 8 бит.

Решение.

Вычислим размер памяти для хранения файла, используя формулу (1.23):

$$I_s = 48\,000 \cdot 8 \cdot 1 \cdot 2 \text{ (бит)} = 48\,000 \cdot 2 \cdot 1 \cdot 8 / 8 \text{ (байт)} = 96\,000 / 1024 = 93,75 \text{ кб.}$$

Ответ: 93,75 кб.

Задача 2. Рассчитайте время звучания моноаудиофайла, если при 16-битном кодировании и частоте дискретизации 32 кГц его объем равен 700 Кбайт.

Решение.

Используя формулу (1.23), запишем выражение для вычисления длительности звукового файла:

$$t = I_s / (F \cdot I \cdot N).$$

Вычислим t :

$$t = (700 \cdot 1024) / (32\,000 \cdot 2 \cdot 1) = 11,2 \text{ секунд.}$$

Ответ: 11,2 секунд.

1.7. Элементы теории множеств

Множество — это совокупность некоторых предметов, объединенных по какому-либо признаку (примеры: множество цифр двоичной системы счисления, множество компьютеров в классе и т. д.). Предметы, из которых состоит множество, называются его *элементами*. Множество может состоять только из одного элемента или вовсе не содержать элементов. Множество, не содержащее элементов, называется *пустым* и обозначается символом \emptyset .

Множества, как правило, обозначаются прописными буквами латинского алфавита. Элементы множеств обозначаются строчными буквами латинского алфавита.

Обычно множество можно задать, если указать:

- 1) все его возможные элементы, например: $Z = \{\text{яблоко, апельсин, лимон}\}$;
- 2) признак (характеристическое свойство), которым обладают все элементы данного множества и который не имеют предметы, не входящие в рассматриваемое множество. Для обозначения характеристического свойства используется соответствующая запись, например: $A = \{x \in \mathbb{R} \mid x / 2 = 0\}$ — множество четных чисел.

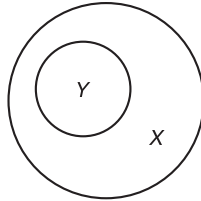
Принадлежность предмета некоторому множеству обозначают с помощью символа \in . И наоборот, символ \notin указывает, что предмет не принадлежит множеству. Например, запись $3 \in \{1, 2, 3\}$ означает, что 3 принадлежит множеству $\{1, 2, 3\}$, а запись $4 \notin \{1, 2, 3\}$ — что 4 не принадлежит указанному множеству.

Множества, состоящие из одних и тех же элементов, называют *равными* и записывают это следующим образом: $X = Y$.

Для представления множеств, отношений между множествами и результатов операций над ними удобно пользоваться *диаграммами Эйлера–Венна (кругами Эйлера)*. При этом множества изображаются на плоскости в виде замкнутых кругов, а элементы множества — это точки внутри соответствующих кругов.

Множество Y , состоящее из некоторых элементов данного множества X и только из них, называется *подмножеством* (частью) множества X . Множество Y называют подмножеством множества X , если любой элемент множества Y является элементом множества X : $Y \subset X$. Иначе говоря, если любой элемент множества Y принадлежит также и множеству X , то множество Y называется подмножеством множества X .

Пусть для обозначения количества элементов в множестве X используется запись N_X , которая обозначает, что в множество X входит N элементов. Если $Y \subset X$, то $N_X \geq N_Y$.

Рис. 1.6. $Y \subset X$

Универсальное множество — это самое большое множество, содержащее в себе все рассматриваемые множества. На диаграмме Эйлера–Венна универсальное множество обозначают в виде прямоугольника и буквы U .



Рис. 1.7. Универсальное множество

Пересечением множеств X и Y называется множество Z , состоящее из элементов, принадлежащих как множеству X , так и множеству Y .

Если множества X и Y не содержат одинаковых элементов, т. е. не пересекаются ($X \cap Y = \emptyset$), то $N_{X \cup Y} = N_X + N_Y$.

Если множества содержат одинаковые элементы $N_{X \cap Y}$, то для определения количества элементов в множестве $N_{X \cup Y}$ должна использоваться формула: $N_{X \cup Y} = N_X + N_Y - N_{X \cap Y}$.

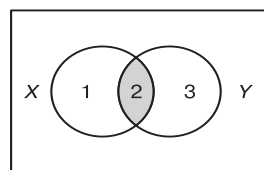


Рис. 1.8. Пересечение двух множеств

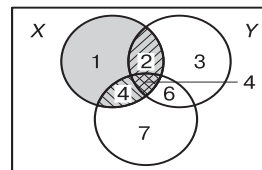


Рис. 1.9. Пересечение трех множеств

Цифрами на рис. 1.8 отмечены множества, образуемые в результате пересечения множеств X и Y , т. е. $N_{X \cap Y} = N_2$, серым цветом отмечен результат пересечения множеств A и B .

Объединением множеств X и Y называется множество Z , которое состоит из всех элементов данных множеств X и Y : $Z = X \cup Y$. На рис. 1.9 результат объединения множеств X и Y обозначен серым цветом.

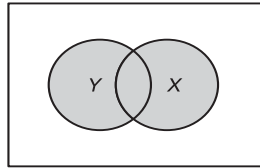


Рис. 1.10. Объединение множеств

Разностью множеств X и Y называется множество элементов Z , принадлежащих множеству X , но не принадлежащих множеству Y (на рис. 1.11 разность множеств X и Y выделена серым цветом): $Z = X \setminus Y$.

Разность между универсальным множеством U и множеством X называется дополнением множества X : $X' = U \setminus X$.

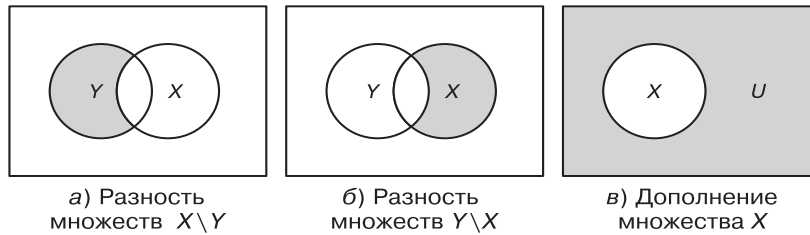


Рис. 1.11

Некоторые операции над множествами показаны на рис. 1.12.

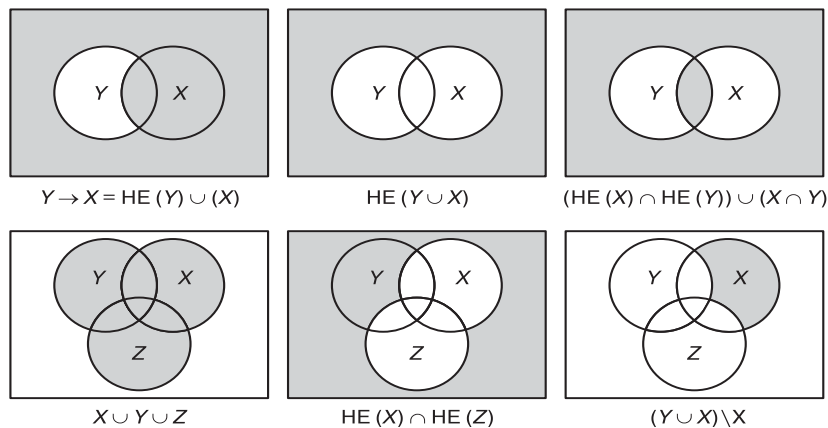
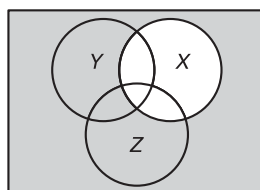


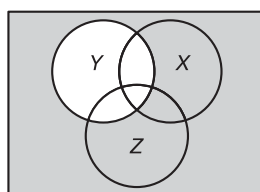
Рис. 1.12. Простейшие операции над множествами

Задача. Изобразите с помощью кругов Эйлера множество $(X' \setminus Y') \cup (Y \cap Z)$.

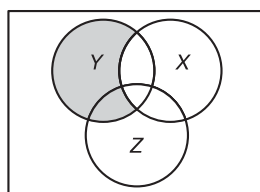
Решение.



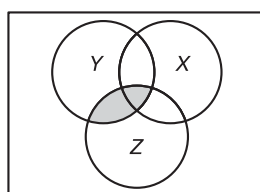
1) X'



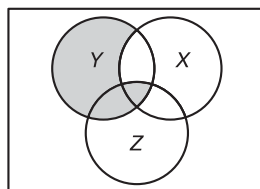
2) Y'



3) $(X' \setminus Y')$



4) $(Y \cap Z)$



5) $(X' \setminus Y') \cup (Y \cap Z)$

1.8. Комбинаторика

Комбинаторика — область математики, в которой рассматриваются задачи о тех или иных комбинациях объектов.

При решении задач, связанных с вычислениями различных комбинаций объектов, используются такие понятия комбинаторики, как размещения, сочетания и перестановки.

Размещениями называют комбинации, составленные из n различных элементов по m элементов ($m < n$), которые отличаются либо составом элементов, либо их порядком.

Существуют формулы для вычисления количества всех возможных размещений без повторений:

$$A_n^m = n! / (n - m)!,$$

а также количества всех возможных размещений с повторениями:

$$A_n^m = n^m.$$

Задача 1. Сколькими способами можно выбрать 9 чисел из 12 без повторений?

Решение.

Для решения этой задачи воспользуемся формулой для подсчета всех возможных размещений без повторений, учитывая, что $m = 9$, а $n = 12$:

$$A_{12}^9 = 12! / (12 - 9)! = 12! / 3! = 12 \cdot 11 \cdot 10 \cdot 9 \cdot 8 \cdot 7 \cdot 6 \cdot 5 \cdot 4 = 79833600.$$

Ответ: 79833600.

Перестановками называют комбинации элементов, состоящие из одних и тех же n различных элементов и отличающиеся только порядком их расположения. Перестановки — это частный случай размещений. Количество всех возможных перестановок элементов множества без повторений вычисляется по формуле:

$$P_n = n!,$$

с повторениями — по формуле:

$$P_n(m_1, m_2, \dots, m_k) = n! / (m_1! \cdot m_2! \cdot \dots \cdot m_k!), \quad n = m_1 + m_2 + \dots + m_k.$$

Задача 2. Сколько существует вариантов размещения пяти целых чисел в массиве из пяти элементов?

Решение.

$$P_5 = 5! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 = 120.$$

Ответ: 120 вариантов.

Сочетаниями называют комбинации, составленные из n различных элементов по m элементов, которые отличаются хотя бы одним элементом. Отличие сочетаний от размещений заключается в том, что в сочетаниях не учитывается порядок элементов.

Количество сочетаний без повторений:

$$C_n^m = n! / (m! \cdot (n - m)!).$$

Количество сочетаний с повторениями:

$$C_n^m = (n + m - 1)! / (m! \cdot (n - 1)!).$$

Задача 3. Сколько существует вариантов выбора 5 чисел из массива, состоящего из 9 элементов?

Решение.

Пять чисел из девяти элементов можно выбрать следующим числом способов:

$$C_9^5 = 9! / (5! \cdot 4!) = (9 \cdot 8 \cdot 7 \cdot 6) / (1 \cdot 2 \cdot 3 \cdot 4) = 126.$$

Ответ: 126 вариантов.

Количества размещений, перестановок и сочетаний связаны между собой равенством:

$$A_n^m = P_m \cdot C_n^m.$$

При решении задач комбинаторики используются следующие правила:

1) *правило суммы* — если некоторый объект A может быть выбран из совокупности объектов m способами, а другой объект B может быть выбран n способами, то выбрать либо A , либо B можно $m + n$ способами;

2) *правило произведения* — если объект A можно выбрать из совокупности объектов m способами и после каждого такого выбора объект B можно выбрать n способами, то пара объектов (A, B) в указанном порядке может быть выбрана $m \cdot n$ способами.

Задача 4. Существует 5 трехзначных чисел и 4 двухзначных. Сколькими способами можно выбрать четыре числа так, чтобы среди них было не больше двух двухзначных?

Решение.

Рассмотрим возможные варианты выбора:

1) выбор 4 трехзначных чисел: $C_5^4 = 5$;

2) выбор трех трехзначных и одного двухзначного: $C_4^1 \cdot C_5^3 = 40$;

3) выбор двух двухзначных и двух трехзначных: $C_4^2 \cdot C_5^2 = 60$.

Таким образом, существует $5 + 40 + 60 = 105$ вариантов выбора.

Ответ: 105 вариантов.

Задача 5. Сколькими различными способами можно записать число, состоящее из 10 цифр, при условии что цифры 1 и 2 не должны стоять рядом?

Решение.

Всего существует $10!$ различных перестановок десяти цифр. Цифры 1 и 2 могут располагаться рядом в двух случаях: «12» и «21». Цифры 1 и 2 стоят рядом в $9!$ вариантах, цифры 2 и 1 стоят рядом также в $9!$ вариантах. Таким образом, мы получим $9! + 9!$ перестановок цифр 1 и 2. Следовательно, число, в котором цифры 1 и 2 не стоят рядом, можно записать $10! - 2 \cdot 9! = 2\,903\,040$ способами.

Ответ: 2 903 040.

Задача 6. В состав проверочной комиссии школьной и городских олимпиад по информатике входят 7 специалистов первой категории и 10 — высшей категории. Нужно выбрать 4 специалиста первой категории для работы в четырех школах и 6 специалистов высшей категории для проверки результатов городской олимпиады. Сколькими способами можно осуществить такой выбор?

Решение.

Одна группа из трех специалистов первой категории может отличаться от другой не только составом специалистов, но и их распределением по школам. В этом случае количество вариантов вычисляется как A_7^4 .

Группа специалистов высшей категории направляется на проверку городской олимпиады, поэтому их порядок в группе не важен. Следовательно, количество вариантов в этом случае — C_{10}^6 .

Для подсчета общего количества вариантов используем правило произведения. Каждый выбор группы учителей первой категории может быть осуществлен при C_{10}^6 вариантах выбора учителей высшей категории. Следовательно, всего существует $A_7^4 \cdot C_{10}^6$ вариантов.

Задача 7. Имеется пять различных цифр. Сколько чисел можно составить из этих цифр, если каждое число состоит из трех цифр, причем две соседние цифры должны быть разными?

Решение.

Если число состоит из трех разных цифр, то одно число от другого может отличаться не только самими цифрами, но и порядком цифр. Это означает, что существует A_5^3 различных чисел, состоящих из трех разных цифр.

Число может состоять из двух цифр, одна из которых входит в число два раза (примеры: 121 или 212). Выбор двух разных цифр можно осуществить C_5^2 способами, и при использовании каждого способа будут получены два различных числа.

$$1) A_5^3 = 5!/(5-3)! = 5!/2! = 5 \cdot 4 \cdot 3 = 60;$$

$$2) C_5^2 = 5!/(2! \cdot 3!) = 5 \cdot 4/(1 \cdot 2) = 10.$$

С учетом правила суммы можно сказать, что всего можно составить $A_5^3 + 2 C_5^2$ различных чисел.

Таким образом, всего существует $60 + 2 \cdot 10 = 80$ чисел.

Ответ: 80.

Задача 8. Сколько существует четырехзначных чисел, в записи которых ровно две шестерки, стоящие рядом?

Решение.

Существует три варианта размещения рядом двух шестерок в четырехзначном числе:

$$1) 66++;$$

$$2) +66+;$$

$$3) ++66,$$

где символ «+» обозначает произвольную цифру числа.

В первом случае две последних цифры не могут быть равны «6», т. е. остается 9 цифр, которые можно записать на место двух последних в первом варианте числа. Это позволяет сгенерировать $9 \cdot 9 = 81$ различных чисел.

Во втором случае первая цифра не может быть равна «0» и не может быть равна «6» (возможно 8 допустимых цифр), а последняя цифра не может быть «6». Это позволяет сгенерировать $8 \cdot 9 = 72$ различных чисел.

В третьем случае обе цифры не могут быть равны «0» или «6» (т. е. возможно 8 допустимых цифр). Это позволяет сгенерировать $8 \cdot 8 = 64$ различных чисел.

В результате мы получаем, что количество четырехзначных чисел, в записи которых ровно две шестерки, стоящие рядом, равно $81 + 72 + 64 = 217$.

Ответ: 217.

Задачи для самостоятельной работы

Методы измерения количества информации

Задача 1. Сколько бит информации содержит сообщение объемом 2^3 мегабайта?

Задача 2. Световое табло состоит из лампочек. Каждая лампочка может находиться в одном из трех состояний («включено», «выключено» или «мигает»). Какое наименьшее количество лампочек должно находиться на табло, чтобы с его помощью можно было передать 20 различных сигналов?

Задача 3. Световое табло состоит из лампочек. Каждая лампочка может находиться в одном из трех состояний («включено», «выключено» или «мигает»). Какое количество сигналов можно передать, используя 4 лампочки?

Задача 4. Световое табло состоит из лампочек. Каждая лампочка может находиться в одном из 4 состояний. Какое наименьшее количество лампочек должно находиться на табло, чтобы с его помощью можно было передать 200 различных сигналов?

Задача 5. Для передачи секретного сообщения из 100 символов использовался код, состоящий из 20 букв. Все буквы кодируются одним и тем же минимально возможным количеством битов. Определите, чему равен информационный объем такого сообщения в битах.

Задача 6. В спичечном коробке лежат целые и сломанные спички. Сломанных спичек в коробке — 9 штук. Сообщение о том, что из коробки достали целую спичку, несет 2 бита информации. Сколько всего спичек в коробке?

Задача 7. В ящике находятся 64 лампочки, среди которых X исправных. Наудачу вынимается одна лампочка. Сообщение «извлечена неисправная лампочка» несет 5 бит информации. Сколько в ящике находится исправных лампочек?

Задача 8. В двух вагонах пассажирского поезда находится 16 пассажиров. Сообщение «пассажир первого вагона уснул» несет 2 бита информации. Сколько пассажиров находится во втором вагоне?

Задача 9. В школе есть три одиннадцатых класса: 11а, 11б, 11в. Каждый день случайно выбирается один из учеников параллели 11-х классов для выполнения обязанностей дежурного. Количество информации, содержащейся в сообщении:

1) «Выбран ученик 11а класса» равно $(1 + \log_2 2)$ бит;

2) «Выбран ученик 11б класса» равно $(1 + \log_2 3)$ бит.

В 11а и 11б классах учатся 10 школьников. Определите количество учеников в 11в классе.

Задача 10. Школьники участвуют в олимпиадах по информатике, математике и физике. Каждый школьник участвует только в одной олимпиаде. Информационный объем сообщения «Школьник не участвует в олимпиаде по математике» равен $(\log_2 3 - 1)$ бит, а сообщения «Школьник участвует в олимпиаде по физике» — $\log_2 3$ бит. Определите количество школьников, принимающих участие в олимпиадах, если известно, что в олимпиаде по информатике участвует 3 школьника.

Процесс передачи информации

Задача 1. Передача данных через ADSL-соединение заняла 3 минуты. За это время был передан файл размером 5720 Кб. Определите минимальную скорость (в Кбит/с), при которой возможна такая передача.

Задача 2. Передача файла размером 3 Мб по каналу связи заняла 35 секунд. Определите скорость передачи данных по каналу связи в Кбит/с.

Задача 3. Скорость передачи данных через ADSL-соединение равна 56 Кбит/с. Передача файла через данное соединение заняла 3 минуты. Определите размер файла в килобайтах.

Задача 4. Информационное сообщение объемом 1,5 Кб передается со скоростью 1024 бит/с. Определите время (в секундах), за которое будет передано данное сообщение.

Задача 5. У Васи есть доступ к сети Интернет по высокоскоростному одностороннему радиоканалу, обеспечивающему скорость получения информации 512 Кбит/с. У Пети нет скоростного доступа в Интернет, но есть возможность получать информацию от Васи по низкоскоростному телефонному каналу со средней скоростью 16 Кбит/с. Петя договорился с Васей, что тот будет скачивать для него данные объемом 10 Мбайт по высокоскоростному каналу и ретранслировать их Пете по низкоскоростному каналу.

Компьютер Васи может начать ретрансляцию данных не раньше, чем им будет получен первый 1 Мб этих данных. Каков минимально возможный промежуток времени (в секундах) с момента начала скачивания Васей данных до полного их получения Петей?

Представление числовой информации

Задача 1. На какую цифру оканчивается запись числа $15A_{16}$ в системе счисления с основанием 8?

Задача 2. На какую цифру оканчивается запись десятичного числа 153 в системе счисления с основанием 16?

Задача 3. На какую цифру оканчивается запись числа $15A_{16}$ в системе счисления с основанием 10?

Задача 4. Определите количество единиц в двоичной записи десятичного числа 250,5.

Задача 5. Чему равно количество значащих нулей в двоичной записи десятичного числа 101_{10} ?

Задача 6. Определите количество единиц в двоичной записи десятичного числа 200,25.

Задача 7. Определите, выполняется ли неравенство $A < C < B$ при $A = 151_8$, $B = 6B_{16}$, $C = 01101010$.

Задача 8. Определите, выполняется ли неравенство $A < C < B$ при $A = 214_8$, $B = 8E_{16}$, $C = 10001101$.

Задача 9. Определите, какое число будет получено из числа $10,001_2$ переносом запятой на 2 разряда влево. Полученное число переведите в десятичную систему счисления.

Задача 10. Определите, какое число будет получено из числа $12,2102_4$ переносом запятой на 3 разряда влево. Полученное число переведите в десятичную систему счисления.

Задача 11. Определите, какое число будет получено из числа $101,24_8$ переносом запятой на 1 разряд вправо. Полученное число переведите в десятичную систему счисления.

Задача 12. Определите, чему равно восьмеричное число $1,1(3)_8$ в системе счисления по основанию 2.

Задача 13. Определите, чему равно четверичное число $3,11(2)_4$ в системе счисления по основанию 8.

Задача 14. Определите, чему равно двоичное число $10,11(100)_2$ в системе счисления по основанию 8.

Задача 15. Определите, чему равно шестнадцатеричное число $1,F(E)_{16}$ в системе счисления по основанию 8.

Задача 16. Какое количество битов содержит двоичное представление числа 1000_{10} ? Ответ определите без перевода данного числа в двоичную систему счисления.

Задача 17. На железнодорожной станции с помощью специального устройства выполняется регистрация номеров железнодорожных вагонов. Каждый номер состоит из 8 цифр и кодируется минимально возможным числом битов, одинаковым для каждого номера. Каков информационный объем сообщения в байтах, записанного устройством после того, как через железнодорожную станцию проследовал состав из 100 вагонов?

Задача 18. В некоторой стране автомобильный номер состоит из 8 символов. В качестве символов используют 5 различных букв и десятичные цифры в любом порядке.

Каждый такой номер в компьютерной программе записывается минимально возможным и одинаковым целым количеством байтов, при этом используется посимвольное кодирование, а все символы кодируются одинаковым и минимально возможным количеством битов.

Определите объем памяти в байтах, отводимый этой программой для записи 40 номеров.

Задача 19. Для передачи по каналу связи сообщения, состоящего только из символов А, Б, В и Г, используется посимвольное кодирование: А — 000, Б — 100, В — 101, Г — 111. Через канал связи передается сообщение ГВАБА. Закодируйте это сообщение данным кодом. Полученную двоичную последовательность переведите в шестнадцатеричный вид.

Задача 20. Для передачи по каналу связи сообщения, состоящего только из символов А, Б, В и Г, используется посимвольное кодирование: А — 01, Б — 11, В — 100, Г — 111. Через канал связи передается сообщение АБВГБ. Закодируйте это сообщение данным кодом. Полученную двоичную последовательность переведите в восьмеричный вид.

Задача 21. Для кодирования букв А, Б, В, Г решили использовать двухразрядные последовательные двоичные числа от 11 до 00 соответственно. Определите, что получится, если таким способом закодировать последовательность символов ГБАВ и записать результат в шестнадцатеричной системе счисления.

Задача 22. Для кодирования букв А, Б, В, Г решили использовать двухразрядные последовательные двоичные числа от 11 до 00 соответственно. Определите, что получится, если таким способом закодировать последовательность символов БВААГБ и записать результат в шестнадцатеричной системе счисления.

Задача 23. Для кодирования букв А, Б, В, Г решили использовать двухразрядные последовательные двоичные числа 00, 10, 01, 11 соответственно. Определите, что получится, если таким способом закодировать последовательность символов АБВАГГБА и записать результат в шестнадцатеричной системе счисления.

Задача 24. Для 5 букв латинского алфавита заданы их двоичные коды (для некоторых букв — из двух битов, а для некоторых — из трех). Эти коды представлены в таблице:

a	b	c	d	e
01	101	00	111	10

Определите набор букв, который закодирован двоичной строкой 101000011101.

Задача 25. Для 5 букв латинского алфавита заданы их двоичные коды (для некоторых букв — из двух битов, а для некоторых — из трех). Эти коды представлены в таблице:

q	e	b	k	l
00	111	101	10	010

Определите набор букв, который закодирован двоичной строкой 00101111101010.

Задача 26. Для 5 букв русского алфавита заданы их двоичные коды (для некоторых букв — из двух битов, а для некоторых — из трех). Эти коды представлены в таблице:

А	К	Н	О	Д
11	00	101	100	01

Определите, корректно ли декодируется сообщение 101100010011.

Задача 27. Для 5 букв русского алфавита заданы их двоичные коды (для некоторых букв — из двух битов, а для некоторых — из трех). Эти коды представлены в таблице:

А	К	Н	О	Д
11	00	101	100	01

Определите, корректно ли декодируется сообщение 110101101000.

Задача 28. На какую цифру оканчивается запись десятичного числа 806 в системе счисления с основанием 8?

Задача 29. В саду 100_q фруктовых деревьев, из них 33_q яблонь, 22_q груш, 16_q слив и 5 вишен. В какой системе счисления подсчитаны количества деревьев?

Задача 30. В корабельный состав флота США входят 167_q боевых кораблей дальней морской зоны. Из них 13_q авианосцев, 26_q крейсеров УРО, 70_q эсминцев УРО, 36_q фрегатов. В какой системе счисления подсчитаны количества кораблей?

Задача 31. Десятичное число 18 записали в системе счисления с основанием q , после чего оно приняло вид 10010. Найти q .

Задача 32. Десятичное число 31 записали в системе счисления с основанием q , после чего оно приняло вид 111. Найти q .

Задача 33. Найти десятичное число x , запись которого в системе счисления с основанием 3 оканчивается на 11 и которое удовлетворяет неравенству $20 < x < 30$.

Задача 34. Найти десятичное число x , запись которого в системе счисления с основанием 3 оканчивается на 11 и которое удовлетворяет неравенству $10 < x < 20$.

Задача 35. Укажите все десятичные числа, не превосходящие 100, запись которых в системе счисления с основанием 5 оканчивается на 22.

Задача 36. Найдите основание системы счисления (p), в которой верно следующее равенство: $2000_p - 300_p = 1500_p$.

Задача 37. Укажите наименьшее основание системы счисления, в которой запись числа 29 трехзначна.

Задача 38. Укажите через запятую в порядке возрастания все основания систем счисления, в которых запись числа 31 оканчивается на 3.

Задача 39. Укажите через запятую в порядке возрастания все основания систем счисления, в которых запись числа 12 оканчивается на 2.

Задача 40. Не переводя непосредственным делением «в столбик» десятичное число 1025 в двоичную систему, определите количество нулей в его двоичном представлении.

Задача 41. Укажите, сколько раз используется цифра 3 при записи чисел 23, 24, 25, ..., 29, 30 в системе счисления с основанием 4.

Задача 42. В непозиционной системе счисления, которая называется *системой остатков*, в качестве оснований выбираются взаимно простые числа (например: $p_1 = 3, p_2 = 5, p_3 = 7$). При этом диапазон однозначного представления чисел равен произведению оснований $D = p_1 \cdot p_2 \cdot p_3$ (от 0 до 104). Любое число в этом диапазоне записывается остатками от целочисленного деления этого числа на выбранные основания. Например, число $A = 19$ запишется в системе остатков с основаниями 3, 5, 7 как $A = (1, 4, 5)$. Определите запись, которая соответствует числу 12, записанному в указанной системе остатков.

Задача 43. Определите наименьшее основание позиционной системы счисления x , при котором $21_x = 14_y$.

Задача 44. Определите наименьшее основание позиционной системы счисления x , при котором $102_x = 516_y$.

Задача 45. Определите наименьшее основание позиционной системы счисления x , при котором $104_x = 566_y$.

Задача 46. С узла связи A на узел связи B передано сообщение CDAB, закодированное с помощью следующей кодовой таблицы:

A	B	C	D
00	01	0	1

В сообщении отсутствуют промежутки, отделяющие одну букву от другой. Определите количество возможных способов прочтения переданного сообщения.

Задача 47. С узла связи A на узел связи B передано сообщение CADBB, закодированное с помощью следующей кодовой таблицы:

A	B	C	D
100	0	101	1

В сообщении отсутствуют промежутки, отделяющие одну букву от другой. Определите количество возможных способов прочтения переданного сообщения.

Задача 48. Шестнадцатеричное четырехзначное число заканчивается цифрой С. Первую цифру переставили в конец числа. Полученное число оказалось на 1905_{16} больше исходного. Определите, чему равно исходное число, записанное в системе счисления по основанию 16.

Задача 49. Шестнадцатеричное четырехзначное число заканчивается цифрой 1. Первую цифру переставили в конец числа. Полученное число оказалось на 4209_{16} больше исходного. Определите, чему равно исходное число, записанное в системе счисления по основанию 16.

Задача 50. Восьмеричное пятизначное число заканчивается цифрой 2. Первую цифру переставили в конец числа. Полученное число оказалось на 35661_8 больше исходного. Определите, чему равно исходное число, записанное в системе счисления по основанию 8.

Задача 51. Сгенерируйте код Хэмминга (7, 4) для комбинации 1011.

Задача 52. Сгенерируйте код Хэмминга (7, 4) для комбинации 1000.

Задача 53. Сгенерируйте код Хэмминга (7, 4) для комбинации 1111.

Задача 54. Получена кодовая последовательность 1110011. Известно, что она закодирована кодом Хэмминга (7, 4) и в ней при передаче был искажен один символ. Определите символ, в котором допущена ошибка.

Задача 55. Получена кодовая последовательность 1110001. Известно, что она закодирована кодом Хэмминга (7, 4) и в ней при передаче был искажен один символ. Определите символ, в котором допущена ошибка.

Задача 56. Получена кодовая последовательность 1111101. Известно, что она закодирована кодом Хэмминга (7, 4) и в ней при передаче был искажен один символ. Определите символ, в котором допущена ошибка.

Задача 57. Определите код Хаффмана для каждого символа, входящего в приведенный ниже текст (без учета кавычек): «abc aad cba caba faa». Учтите, что каждый символ кодируется минимальным количеством битов, определите размер текста до и после выполнения такого кодирования.

Задача 58. Определите код Хаффмана для каждого символа, входящего в приведенный ниже текст (без учета кавычек): «...1230.1022.1230.75438.342». Учтите, что каждый символ кодируется минимальным количеством битов, определите размер текста до и после выполнения такого кодирования.

Задача 59. Определите код Хаффмана для каждого символа, входящего в приведенный ниже текст (без учета кавычек): «m--.asd--mm.+mmm.asd.». Учитывая, что каждый символ кодируется минимальным количеством битов, определите размер текста до и после выполнения такого кодирования.

Представление чисел в прямом, обратном и дополнительном кодах

Задача 1. Запишите в десятичной системе счисления целое число, если дан его дополнительный код: 10001111.

Задача 2. Запишите в десятичной системе счисления целое число, если дан его обратный код: 11000000.

Задача 3. Запишите дополнительный код числа -100 .

Задача 4. Запишите обратный код числа -90 .

Задача 5. Выполните сложение чисел $X = -10011$ и $Y = 101$ в обратном и дополнительном кодах.

Задача 6. Для хранения целого числа со знаком в ЭВМ используется один байт. Сколько единиц содержит внутреннее представление числа -78 , записанного в дополнительном коде?

Задача 7. Для хранения целого числа со знаком в ЭВМ используется один байт. Сколько единиц содержит внутреннее представление числа -55 , записанного в обратном коде?

Задача 8. Какое равенство получится после восстановления двоичных цифр в следующем примере: $100^{**} + ^{*}11 = 1^{*}001$?

Задача 9. Какое равенство получится после восстановления шестнадцатеричных цифр в следующем примере: $F1^{**} + ^{*}23 = ^{*}2E4$?

Представление чисел с фиксированной и плавающей запятой

Задача 1. В двух байтах находится целое положительное число, записанное в формате с фиксированной запятой: 0111000011101010. Какому целому восьмеричному числу соответствует эта запись?

Задача 2. В двух байтах находится целое положительное число, записанное в формате с фиксированной запятой: 0111000011101010. Какому целому шестнадцатеричному числу соответствует эта запись?

Задача 3. Значение переменной A представлено в формате с плавающей точкой в шестнадцатеричной системе счисления: $A = 42F20000_{16}$. Тип переменной A — вещественное число с одинарной точностью. Определите десятичное значение числа A .

Задача 4. Значение переменной A представлено в формате с плавающей точкой в шестнадцатеричной системе счисления: $A = C34E0000_{16}$. Тип переменной A — вещественное число с одинарной точностью. Определите десятичное значение числа A .

Задача 5. Значение переменной A представлено в формате с плавающей точкой в шестнадцатеричной системе счисления: $A = 42D40000_{16}$. Тип переменной A — вещественное число с одинарной точностью. Определите десятичное значение числа A .

Задача 6. Значение переменной A представлено в формате с плавающей точкой в шестнадцатеричной системе счисления: $A = 425C0000_{16}$. Тип переменной A — вещественное число с одинарной точностью. Определите десятичное значение числа A .

Задача 7. Значение переменной A представлено в формате с плавающей точкой в шестнадцатеричной системе счисления: $A = 43FC8000_{16}$. Тип переменной A — вещественное число с одинарной точностью. Определите десятичное значение числа A .

Задача 8. Представьте число $716,4_8$ в формате с плавающей точкой с одинарной точностью. Результат запишите в виде шестнадцатеричного числа.

Задача 9. Представьте число $1001,11_2$ в формате с плавающей точкой с одинарной точностью. Результат запишите в виде шестнадцатеричного числа.

Задача 10. Представьте число $3213,2_4$ в формате с плавающей точкой с одинарной точностью. Результат запишите в виде шестнадцатеричного числа.

Арифметические операции над числами

Задача 1. Вычислите сумму чисел $X = 2D_{16}$ и $Y = 32_8$, результат представьте в двоичной системе счисления.

Задача 2. Вычислите сумму чисел $X = 3C_{16}$ и $Y = 10011110_2$, результат представьте в восьмеричной системе счисления.

Задача 3. Вычислите сумму чисел $X = 201_3$ и $Y = 412_5$, результат представьте в двоичной системе счисления.

Задача 4. Определите разность чисел $F2_{16}$ и 17_8 в четверичной системе счисления.

Задача 5. Определите разность чисел 13_4 и 101_2 в троичной системе счисления.

Задача 6. Определите значение выражения $10_{16} + 10_8 : 10_2$ в шестнадцатеричной системе счисления.

Задача 7. Определите значение выражения $10_8 - 10_{16} : 10_2$ в десятичной системе счисления.

Задача 8. Определите значение выражения $2F_{16} - 10_8 \cdot 10_2$ в двоичной системе счисления.

Задача 9. Определите остаток от деления 58_{16} на 37_8 в десятичной системе счисления.

Задача 10. Определите остаток от деления $2F_{16}$ на 101011_2 в десятичной системе счисления.

Задача 11. Определите остаток от деления 36_8 на $1F_{16}$ в двоичной системе счисления.

Задача 12. Определите сумму чисел $232,3_4$ и $321,2_5$. Ответ запишите в десятичной системе счисления.

Задача 13. Определите произведение чисел $31,3_6$ и $21,2_5$. Ответ запишите в десятичной системе счисления.

Задача 14. Определите разность чисел A, A_{16} и $101,1_2$. Ответ запишите в десятичной системе счисления.

Кодирование текстовой информации

Задача 1. Считая, что каждый символ кодируется одним байтом, определите информационный объем сообщения в байтах: «Единица измерения информации называется битом.» (кавычки входят в текст сообщения).

Задача 2. В кодировке CP-1251 на кодирование одного символа отводится 8 битов. Определите в байтах информационный объем сообщения высказывания Федора Достоевского: «Красота спасет мир.» (кавычки входят в текст сообщения).

Задача 3. В кодировке Unicode на кодирование одного символа отводится 2 байта. Определите в битах информационный объем высказывания Федора Достоевского: «Богатство, грубость наслаждений порождают лень, а лень порождает рабов.» (кавычки также входят в сообщение).

Задача 4. В кодировке Unicode на кодирование одного символа отводится 16 битов. Определите в байтах информационный объем сообщения из 10 символов, записанного в кодировке Unicode.

Задача 5. Автоматическое устройство осуществило перекодировку информационного сообщения на русском языке, первоначально записанного в 16-битном коде Unicode, в 8-битную кодировку CP-1251. При этом информационное сообщение уменьшилось на 320 битов. Определите длину сообщения в символах.

Задача 6. Автоматическое устройство осуществило перекодировку информационного сообщения на русском языке, первоначально записанного в 8-битной кодировке КОИ-8, в 16-битный код Unicode. При этом информационное сообщение увеличилось на 560 бит. Определите длину сообщения в символах.

Задача 7. Объем сообщения равен 4,5 Кб. Известно, что данное сообщение содержит 2304 символов. Определите мощность алфавита.

Задача 8. Объем информационного сообщения равен 16 384 бита. Определите объем этого сообщения в килобайтах.

Задача 9. В таблице ниже представлена часть кодовой таблицы ASCII:

Символ	1	5	A	B	Z	a	b
Десятичный код	49	53	65	66	90	97	98
Шестнадцатеричный код	31	35	41	42	5A	61	62

Определите шестнадцатеричный код символа «z».

Задача 10. Дан текст, состоящий из 100 символов. Известно, что символы берутся из таблицы размером 8×16 и кодируются минимальным количеством битов. Определите информационный объем текста в битах.

Задача 11. Сколько секунд потребуется модему, передающему сообщения со скоростью 8 Кбит/с, чтобы передать файл, содержащий 2048 символов? Каждый символ кодируется 4 битами.

Задача 12. Декодировать текст «68 6F 6D 65» с помощью таблицы ASCII-кодов.

Задача 13. Код символа «О» в таблице кодировки CP-866 равен 79. Определите, какое слово записано с помощью последовательности кодов: 83 80 79 82 84, не используя таблицу кодировки.

Задача 14. Закодировать слово «Информатика» в кодировке CP-1251.

Кодирование графической информации

Задача 1. Для хранения растрового изображения размером 64×64 пикселя отвели 1024 байт памяти. Каково максимально возможное число цветов в палитре изображения?

Задача 2. Разрешение экрана монитора — 1024×768 пикселей, глубина цвета — 24 бита. Каков необходимый объем видеопамати (в мегабайтах) для данного графического режима?

Задача 3. Какой объем памяти (в килобайтах) необходим для хранения 16-цветного растрового графического изображения размером 32×128 пикселей?

Задача 4. Сколько секунд потребуется модему, передающему информацию со скоростью 32 000 бит/с, чтобы передать 64-цветное растровое изображение размером 800×600 пикселей при условии, что в одном байте закодировано максимально возможное целое число пикселей?

Задача 5. Сколько секунд потребуется модему, передающему информацию со скоростью 32 000 бит/с, чтобы передать 32-цветное растровое изображение размером 640×480 пикселей при условии, что в одном байте закодировано максимально возможное целое число пикселей?

Задача 6. Для кодирования цвета фона web-страницы используется атрибут `bgcolor = "#XXXXXX"`, где в кавычках задаются шестнадцатеричные значения интенсивности цветовых компонентов в 4-битной RGB-модели. Какой цвет будет у страницы, заданной тегом `<body bgcolor = "#FFFF00">`?

Задача 7. Для кодирования цвета фона web-страницы используется атрибут `bgcolor = "#XXXXXX"`, где в кавычках задаются шестнадцатеричные значения интенсивности цветовых компонентов в 24-битной RGB-модели. Какой цвет будет у страницы, заданной тегом `<body bgcolor = "#070707">`?

Задача 8. В процессе преобразования растрового графического файла количество цветов уменьшилось с 256 до 16. Во сколько раз уменьшился информационный объем файла?

Задача 9. В процессе преобразования растрового графического файла количество цветов уменьшилось с 256 до 4. Во сколько раз уменьшился информационный объем файла?

Задача 10. Цветной сканер имеет разрешение 512×512 точек/дюйм. Объем памяти, занимаемой отсканированным изображением размером 8×8 дюймов, составляет 8 Мб. Определите глубину представления цвета сканером в битах.

Задача 11. Цветной сканер имеет разрешение 256×512 точек/дюйм. Объем памяти, занимаемой отсканированным изображением размером 4×4 дюйма, составляет 3 Мб. Определите глубину представления цвета сканером в битах.

Кодирование звуковой информации

Задача 1. Определите объем памяти (в килобайтах) для хранения цифрового аудиофайла, содержащего стереозапись, время звучания которой составляет две секунды при частоте дискретизации 16 кГц и разрешении 16 бит.

Задача 2. Определите объем памяти (в килобайтах) для хранения цифрового аудиофайла, содержащего монозапись, время звучания которой составляет четыре секунды при частоте дискретизации 8 кГц и разрешении 8 бит.

Задача 3. Рассчитайте время звучания моноаудиофайла, если при 32-битном кодировании и частоте дискретизации 32 кГц его объем равен 750 Кб.

Задача 4. Рассчитайте время звучания стереозаписи, если при 8-битном кодировании и частоте дискретизации 48 кГц ее объем равен 1500 Кб.

Задача 5. Определите частоту дискретизации звукового моноаудиофайла, если при 32-битном кодировании и длительности четырех секунд его объем равен 125 Кб.

Задача 6. Определите частоту дискретизации звукового стереоаудиофайла, если при 16-битном кодировании и длительности восьми секунд его объем равен 500 Кб.

Задача 7. Определите время передачи цифрового аудиофайла, содержащего стереозапись, время звучания которой составляет четыре секунды при частоте дискретизации 16 кГц и разрешении 16 бит. Скорость передачи файла — 100 Кб/с.

Задача 8. Определите время передачи цифрового аудиофайла, содержащего монозапись, время звучания которой составляет восемь секунд при частоте дискретизации 8 кГц и разрешении 8 бит. Скорость передачи файла — 100 Кб/с.

Комбинаторика

Задача 1. Сколько существует четырехзначных чисел, в записи которых имеются две цифры семь?

Задача 2. Сколько существует четырехзначных чисел, состоящих из различных нечетных цифр?

Задача 3. Сколько существует четырехзначных чисел, в которых три цифры четные?

Задача 4. Сколько существует четырехзначных чисел, в записи которых есть хотя бы две четные цифры?

Задача 5. Сколько существует четырехзначных чисел, составленных из разных цифр, в записи которых есть две четные цифры?

Задача 6. Сколько существует четырехзначных чисел, которые делятся на два?

Задача 7. Сколько существует четырехзначных чисел, в записи которых все цифры четные и хотя бы одна из них цифра 4?

Задача 8. Сколько существует четырехзначных чисел, не превышающих 5000, в которых имеются ровно две цифры 5?

Задача 9. Сколько существует четырехзначных чисел, в записи которых все цифры разные и есть цифра 1?

Задача 10. Сколько существует четырехзначных чисел, оканчивающихся на 4 или на 6?

Задача 11. Сколькими способами можно расставить на полке семь книг, если две определенные книги должны стоять рядом?

Задача 12. Из цифр 1, 2, 3, 4, 5 составляются всевозможные числа, каждое из которых содержит не менее трех цифр. Сколько таких чисел можно составить, если повторения цифр в числах запрещены?

Задача 13. В автомашине 7 мест. Сколькими способами семь человек могут разместиться в этой машине, если занять место водителя могут только трое из них?

Задача 14. Сколькими способами можно выбрать 3 карты из 20?

Задача 15. Сколько четырехбуквенных комбинаций можно образовать из букв слова «сапфир»?

Задача 16. Сколько вариантов расписания пяти уроков можно составить при выборе из 11 дисциплин?

Задача 17. Сколькими способами можно расставить на полке семь книг, если из них две определенные книги не должны стоять рядом?

Задача 18. Сколькими способами 8 человек могут встать в очередь к театральной кассе?

Теория множеств

Задача 1. Изобразите при помощи кругов Эйлера множество $(X \cup Z) \setminus Y$.

Задача 2. Изобразите при помощи кругов Эйлера множество $(X \cap Y) \cap Z'$.

Задача 3. Изобразите при помощи кругов Эйлера множество $(Z \cap (Q \cup \text{НЕ}(Y))) \cup (X \cap (Y \cup Z))$.

Задача 4. Изобразите при помощи кругов Эйлера множество $(X \mid Y) \cap (Q \cap Z)$.

Задача 5. Изобразите при помощи кругов Эйлера множество $\text{НЕ}(\text{НЕ}(X) \cup \text{НЕ}(Y)) \cup \text{НЕ}(\text{НЕ}(Q) \cup \text{НЕ}(Z))$.

Задача 6. Заданы два множества: $A = \{(x, y) \in \mathbb{R}^2 \mid x^2 + y^2 \leq 9\}$ и $B = \{x \in \mathbb{R} \mid x \geq 0\}$. Графически изобразите их пересечение.

Задача 7. Заданы четыре множества: $A = \{x \in \mathbb{R} \mid x \geq -2\}$, $B = \{x \in \mathbb{R} \mid x \leq 2\}$, $C = \{y \in \mathbb{R} \mid y \leq 2\}$ и $D = \{y \in \mathbb{R} \mid y \geq -2\}$. Графически изобразите их пересечение.

Задача 8. Заданы два множества: $A = \{(x, y) \in \mathbb{R}^2 \mid (x + 1)^2 + y^2 \leq 1\}$ и $C = \{(x, y) \in \mathbb{R}^2 \mid x^2 + (y + 1)^2 \leq 1\}$. Графически изобразите их пересечение.

Задача 9. Заданы два множества: $A = \{(x, y) \in \mathbb{R}^2 \mid x^2 + y^2 \leq 1\}$ и $C = \{x \in \mathbb{R} \mid y \leq x\}$. Графически изобразите множество $C \setminus A$.

Глава 2

Алгебра логики

2.1. Основные понятия алгебры логики

Алгебра логики — раздел математики, изучающий высказывания, рассматриваемые с точки зрения их логических значений (истинности или ложности), и логические операции над ними. *Логическое высказывание* представляет собой любое повествовательное предложение, в отношении которого можно однозначно сказать, истинно оно или ложно. Для построения новых высказываний из уже заданных в логике используются следующие слова и словосочетания: «не», «и», «или», «если ..., то», «тогда и только тогда». Такие слова и словосочетания называются *логическими связками*.

Высказывания, образованные из других высказываний с помощью логических связок, называются *составными*. Высказывания, не являющиеся составными, называются *элементарными*.

Например, из элементарных высказываний «3 больше 2» и «3 меньше 5» при помощи связки «и» можно получить составное высказывание: «3 больше 2 и меньше 5».

Для обращения к логическим высказываниям им назначают имена. Пусть через A обозначено высказывание «3 больше 2», а через B — высказывание «3 меньше 5». Тогда составное высказывание «3 больше 2 и меньше 5» можно записать как « A и B », где A , B — *логические переменные*, которые могут принимать только два значения — «истина» или «ложь» («1» или «0»).

Каждая логическая связка рассматривается как операция над логическими высказываниями и имеет свое название и обозначение:

1) *отрицание (инверсия)* — выражается словом «не» и обозначается чертой над высказыванием (\bar{A}) или знаком \neg ($\neg A$);

2) *конъюнкция (логическое умножение)* — выражается связкой «и» и обозначается одним из следующих знаков: \cdot , \wedge , $\&$;

3) *дизъюнкция (логическое сложение)* — выражается связкой «или» и обозначается одним из следующих знаков: $+$, \vee («включающее или»);

4) *сложение по модулю 2* — выражается связкой «либо ... либо ...» и обозначается одним из следующих знаков: $\hat{}$, \oplus («исключающее или»);

5) *импликация* — выражается связками «если ..., то», «из ... следует», «... влечет ...» и обозначается знаком \rightarrow ;

6) *эквиваленция (двойная импликация, равносильность)* — выражается связками «тогда и только тогда», «необходимо и достаточно», «... равносильно ...» и обозначается знаками \leftrightarrow , \equiv , \sim .

Импликацию можно выразить через дизъюнкцию и отрицание:

$$A \rightarrow B = \overline{A} \vee B. \quad (2.1)$$

Эквиваленцию можно записать через отрицание, дизъюнкцию и конъюнкцию:

$$A \leftrightarrow B = (\overline{A} \vee B) \wedge (\overline{B} \vee A) = A \wedge B \vee \overline{A} \wedge \overline{B}. \quad (2.2)$$

Для каждой логической операции существует *таблица истинности*, которая показывает соответствие между всеми возможными наборами значений входных переменных и результатом применения к данным значениям указанной логической операции (см. табл. 2.1 — таблицу истинности для описанных выше операций).

Таблица 2.1

Таблица истинности логических операций

A	B	$A \wedge B$	$A \vee B$	$A \oplus B$	$A \rightarrow B$	$A \leftrightarrow B$	\overline{A}
0	0	0	0	0	1	1	1
0	1	0	1	1	1	0	1
1	0	0	1	1	0	0	0
1	1	1	1	0	1	1	0

Порядок выполнения логических операций задается скобками. Для уменьшения числа скобок принято считать, что сначала выполняется операция отрицания («не»), затем конъюнкция («и»), после конъюнкции — дизъюнкция («или»), а в последнюю очередь — импликация и эквиваленция. Например, запись $A \vee B \wedge \overline{C}$ эквивалентна формуле $A \vee (B \wedge \overline{C})$.

2.2. Логические выражения и их преобразование

С помощью логических переменных и символов логических операций любое высказывание можно заменить *логической формулой*:

- 1) всякая логическая переменная и символы «истина» (1) и «ложь» (0) — формулы;
- 2) если A и B — формулы, то A , $A \wedge B$, $A \vee B$, $A \rightarrow B$, $A \leftrightarrow B$ — также формулы;
- 3) никаких других формул в алгебре логики нет.

Существуют следующие виды логических формул:

1) *выполнимая формула* — формула, которая для одних сочетаний значений входящих в нее переменных принимает значение «истина», а для других — значение «ложь». Например, формула $A \vee B$ является выполнимой (см. табл. 2.2);

2) *тождественно истинная формула* — формула, которая принимает значение «истина» при любых значениях входящих в нее переменных. Например, формула $A \vee \bar{A}$ является тождественно истинной (см. табл. 2.2), т. е. $A \vee \bar{A} = 1$. Тождественно истинные формулы также называют *тавтологиями*;

3) *тождественно ложная формула* — формула такого вида принимает значение «ложь» при любых значениях входящих в нее логических переменных. Например, $A \wedge \bar{A}$ (см. табл. 2.2), т. е. $A \wedge \bar{A} = 0$. Тождественно ложные формулы также называют *противоречиями*.

Таблица 2.2

Таблица истинности формул

A	B	\bar{B}	$A \vee B$	$A \vee \bar{A}$	$A \wedge \bar{A}$
0	0	1	0	1	0
0	1	1	1	1	0
1	0	0	1	1	0
1	1	0	1	1	0

Часто при решении логических задач требуется упростить логическую формулу или привести ее к определенному виду. Для этого используются *равносильные преобразования логических формул*.

Под *упрощением* формулы, не содержащей операций импликации и эквивалентности, понимают равносильное преобразование, приводящее к формуле, которая либо содержит (по сравнению с исходной) меньшее число операций конъюнкции и дизъюнкции и не содержит отрицаний неэлементарных формул (т. е. формул, которые не являются составными), либо содержит меньшее число вхождений переменных. При упрощении логических формул используются следующие законы алгебры логики:

Переместительный:	$A \vee B = B \vee A,$ $A \wedge B = B \wedge A;$
Сочетательный:	$A \vee (B \vee C) = (A \vee B) \vee C,$ $A \wedge (B \wedge C) = (A \wedge B) \wedge C;$
Распределительный:	$A \wedge (B \vee C) = A \wedge B \vee A \wedge C,$ $A \vee (B \wedge C) = (A \vee B) \wedge (A \vee C);$
Правила де Моргана:	$\overline{A \vee B} = \overline{A} \wedge \overline{B},$ $\overline{A \wedge B} = \overline{A} \vee \overline{B};$
Идемпотенции:	$A \vee A = A,$ $A \wedge A = A;$
Поглощения:	$A \vee (A \wedge B) = A,$ $A \wedge (A \vee B) = A;$
Склеивания:	$(A \wedge B) \vee (\overline{A} \wedge B) = B,$ $(A \vee B) \wedge (\overline{A} \vee B) = B;$
Операция переменной с ее инверсией:	$A \vee \overline{A} = 1,$ $A \wedge \overline{A} = 0;$
Операции с константами:	$A \vee 0 = A, \quad A \vee 1 = 1,$ $A \wedge 0 = 0, \quad A \wedge 1 = A;$
Двойного отрицания:	$\overline{\overline{A}} = A.$

Примеры упрощения логических формул:

1) $\overline{\overline{A \wedge (\overline{A \wedge B})}} = \{\text{правило де Моргана}\} = \overline{\overline{A} \vee (\overline{A \vee B})} = \{\text{закон двойного отрицания}\} = A \vee (\overline{A \wedge B}) = \{\text{распределительный закон}\} = (A \vee A) \wedge (A \vee B) = 1 \wedge (A \vee B) = A \vee B;$

2) $(A \vee B) \wedge (\overline{A \vee B}) \wedge (B \vee C) \wedge (\overline{B \vee C}) \wedge (A \vee C) = \{\text{закон склеивания}\} = A \wedge C \wedge (A \vee C) = \{\text{распределительный закон}\} = A \wedge (C \wedge A \vee C \wedge C) = \{\text{закон идемпотенции}\} = A \wedge (C \wedge A \vee C) = \{\text{распределительный закон}\} = A \wedge C \wedge A \vee A \wedge C = \{\text{закон идемпотенции}\} = A \wedge C \vee A \wedge C = \{\text{закон идемпотенции}\} = A \wedge C;$

3) $(A \rightarrow B) \wedge (\overline{A \wedge B}) \vee (\overline{A \wedge B}) = \{\text{раскрытие импликации}\} = (\overline{A \vee B}) \wedge (\overline{A \wedge B}) \vee (\overline{A \wedge B}) = \{\text{распределительный закон}\} = \overline{A \wedge A \vee A \wedge B \vee B \wedge A \vee B \wedge B \vee (A \wedge B)} = \{\text{операция переменной с ее инверсией}\} = 0 \vee \overline{A \wedge B \vee B \wedge A \vee B \vee A \wedge B} = \{\text{закон идемпотенции}\} = A \wedge B \vee A \wedge B \vee B = \{\text{распределительный закон}\} = B \wedge (A \vee A \vee 1) = \{\text{операция переменной с ее инверсией}\} = B \wedge 1 = \{\text{операции с константами}\} = B.$

Для проверки правильности упрощения можно построить таблицы истинности исходной и преобразованной логической формулы, а затем сравнить их. Эквивалентность результирующих значений, записанных в таблице для каждой формулы, гарантирует верность выполненных преобразований.

Задача 1. Упростите логическое выражение $\neg(A \vee (\neg B \wedge A) \wedge C)$.

Решение.

Упростим выражение, используя закон де Моргана:

$$\begin{aligned}\neg(A \vee (\neg B \wedge A) \wedge C) &= \neg A \wedge \neg((\neg B \wedge A) \wedge C) = \neg A \wedge \neg((\neg B \wedge A) \wedge C) = \\ &= \neg A \wedge (\neg(\neg B \wedge A) \vee \neg C) = \neg A \wedge ((\neg\neg B \vee \neg A) \vee \neg C) = \neg A \wedge (B \vee \neg A \vee \\ &\vee \neg C) = \neg A \wedge (\neg A \vee (B \vee \neg C)) = \{\text{закон поглощения}\} = \neg A.\end{aligned}$$

Ответ: $\neg(A \vee (\neg B \wedge A) \wedge C) = \neg A$.

Задача 2. Определите значение высказывания $\neg((X > 5) \rightarrow (X > 6))$ при $X = 6$.

Решение.

1-й способ.

Подставим $X = 6$ в заданное высказывание и вычислим его значение:

$$\neg((6 > 5) \rightarrow (6 > 6)) = \neg(1 \rightarrow 0) = \neg 0 = 1.$$

2-й способ.

Упростим исходное высказывание:

1) применим к нему формулу (2.1) для раскрытия импликации:
 $\neg(\neg((X > 5) \rightarrow (X > 6)))$;

2) используем правило де Моргана: $\neg\neg(X > 5) \wedge \neg(X > 6)$;

3) используем правило двойного отрицания: $(X > 5) \wedge \neg(X > 6)$.

Получим, что $\neg((X > 5) \rightarrow (X > 6)) = (X > 5) \wedge \neg(X > 6)$.

Подставим $X = 6$ в полученное высказывание: $(6 > 5) \wedge \neg(6 > 6) = 1$.

Ответ: данное высказывание истинно.

Задача 3. Определите, при каких значениях переменных P и Q выражение $(Q \vee P) \vee Q \wedge \neg P \wedge \neg Q$ ложно.

Решение.

Упростим исходное выражение:

$$(Q \vee P) \vee Q \wedge \neg P \wedge \neg Q = \{Q \wedge \neg Q = 0\} = (Q \vee P) \vee 0 = Q \vee P.$$

Полученное выражение ложно при $P = \text{false}$, $Q = \text{false}$ ($P = 0$, $Q = 0$).

Ответ: $P = 0$, $Q = 0$.

Задача 4. Определите значение высказывания:

$\neg(\text{Первая буква имени гласная} \rightarrow \text{Четвертая буква имени согласная})$

для имени «Федор» [1].

Решение.

1-й способ.

Введем для высказываний следующие обозначения:

- 1) A = Первая буква имени гласная;
- 2) B = Четвертая буква имени согласная.

Перепишем исходное выражение с учетом введенных обозначений:

$$\neg(A \rightarrow B).$$

Для имени «Федор» первая буква имени — согласная, а четвертая — гласная, следовательно, и первое, и второе высказывания — ложные, т. е. $A = 0$, $B = 0$. Подставим полученные значения в выражение:

$$\neg(0 \rightarrow 0) = \neg(1) = 0.$$

Следовательно, для рассматриваемого имени высказывание ложно.

2-й способ.

Введем для высказываний следующие обозначения:

- 1) A = Первая буква имени гласная;
- 2) B = Четвертая буква имени согласная.

Используя эти обозначения, перепишем исходное высказывание:

$$\neg(A \rightarrow B).$$

Упростим полученное высказывание:

- 1) используем формулу (2.1) для раскрытия импликации: $\neg(\neg A \vee B)$;
- 2) применяем правило де Моргана: $\neg\neg A \wedge \neg B$;
- 3) используем закон двойного отрицания и записываем конечное выражение: $A \wedge \neg B$.

Учитывая введенные обозначения, запишем исходное высказывание в виде: «Первая буква имени гласная \wedge Четвертая буква имени согласная».

В имени «Федор» первая буква имени — согласная, а четвертая — гласная. Следовательно, данное имя не удовлетворяет указанным условиям и для него исходное высказывание будет ложным.

Ответ: высказывание ложно.

Задача 5. Определите значение высказывания $(\sin(x) > \pi) \vee \vee (y = x \bmod 4) \wedge (x \operatorname{div} y < 2)$ при $x = 3$, $y = 1$.

Решение.

Операция \bmod используется для вычисления остатка от деления, а операция div — для выполнения целочисленного деления.

Выражение $\sin(x) > \pi$ тождественно ложно, так как не существует такого угла, синус которого больше π . Поэтому исходное выражение можно записать в виде:

$$(y = x \bmod 4) \wedge (x \operatorname{div} y < 2).$$

Вычислим значение данного выражения при $x = 3, y = 1$:

$$(1 = 3 \bmod 4) \wedge (3 \operatorname{div} 1 < 2) = 0 \wedge 0 = 0 \text{ (так как } 3 \bmod 4 = 3).$$

Таким образом, выражение $(y = x \bmod 4) \wedge (x \operatorname{div} y < 2)$ при $x = 3, y = 1$ ложно.

Ответ: высказывание ложно.

Задача 6. Каково наибольшее целое положительное число X , при котором высказывание: $((X + 2) \cdot X + 4 > 0) \rightarrow (X \cdot X > 17)$ будет ложным?

Решение.

Применим некоторые преобразования к данному выражению:

$$\begin{aligned} ((X + 2) \cdot X + 4 > 0) \rightarrow (X \cdot X > 17) &= ((X^2 + 4 \cdot X + 2) > 0) \rightarrow \\ &\rightarrow (X^2 > 17) = ((X + 2)^2 > 0) \rightarrow (X^2 > 17). \end{aligned}$$

Введем обозначения: $A = ((X + 2)^2 > 0), B = (X^2 > 17)$. Перепишем исходное высказывание с учетом введенных обозначений: $A \rightarrow B$.

Решим неравенства:

$$1) A: (X + 2)^2 > 0 \Rightarrow X \in (\infty; -2) \cup (-2; \infty);$$

$$2) B: X^2 > 17 \Rightarrow |X| > \sqrt{17} \Rightarrow X \in (\infty; -\sqrt{17}) \cup (-\sqrt{17}; \infty).$$

Заданное высказывание будет ложным, когда B — ложно, а A — истинно (см. таблицу истинности импликации).

Неравенство A истинно при $X \in (\infty; -2) \cup (-2; \infty)$, а B — ложно при $X \in [-\sqrt{17}; -\sqrt{17}]$. Отобразим это на рис. 2.1.

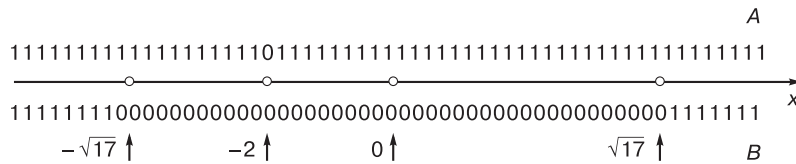


Рис. 2.1. Области истинности и ложности выражения

На рис. 2.1 единицами и нулями обозначены области истинности и ложности выражений A и B . Поэтому исходное выражение будет ложно при $X \in [-\sqrt{17}; -2) \cup (-2; -\sqrt{17}]$.

В задаче требуется найти наибольшее целое положительное число, при котором высказывание будет ложным. Таким числом является первое целое число, меньшее $\sqrt{17}$, т. е. число 4.

Ответ: 4.

Задача 7. A, B, C — целые числа, для которых истинно высказывание:

$$\neg(A \equiv B) \wedge ((B < A) \rightarrow (2C > A)) \wedge ((A < B) \rightarrow (A > 2C)).$$

Чему равно A , если $C = 9, B = 20$?

Решение.

Способ решения этой задачи заключается в следующем:

- 1) упростим исходное выражение;
- 2) подставим значения переменных в упрощенное выражение;
- 3) рассмотрим выражение по частям и выделим в нем ложные компоненты;
- 4) сделаем вывод относительно величины искомой переменной.

Применим к выражению $(B < A) \rightarrow (2C > A)$ формулу (2.1):

$$(B < A) \rightarrow (2C > A) = \overline{(B < A)} \vee (2C > A).$$

Избавимся от отрицания выражения $B < A$, поменяв знак у неравенства:

$$\overline{(B < A)} \vee (2C > A) = (B \geq A) \vee (2C > A).$$

Подставив значения переменных в выражение, получим:

$$(B \geq A) \vee (2C > A) = (20 \geq A) \vee (2 \cdot 9 > A) = (A \leq 20) \vee (A < 18).$$

Преобразуем выражение $(A < B) \rightarrow (A > 2C)$, используя формулу (2.1):

$$(A < B) \rightarrow (A > 2C) = \overline{(A < B)} \vee (A > 2C).$$

Избавимся от отрицания выражения $A < B$:

$$(A \geq B) \vee (A > 2C) = (A \geq 20) \vee (A > 18).$$

Запишем исходное выражение с учетом выполненных вычислений:

$$\begin{aligned} & \neg(A \equiv B) \wedge ((B < A) \rightarrow (2C > A)) \wedge ((A < B) \rightarrow (A > 2C)) = \\ & = (A \equiv 18) \wedge ((A \leq 20) \vee (A < 18)) \wedge ((A \geq 20) \vee (A > 18)). \end{aligned}$$

Применим к полученному выражению распределительный закон:

$$\begin{aligned} & (A \equiv 20) \wedge ((A \leq 20) \vee (A < 18)) \wedge ((A \geq 20) \vee (A > 18)) = \\ & = (A \equiv 20) \wedge (A \leq 20 \wedge A \geq 20) \vee (A \leq 20 \wedge A < 18) \vee \\ & \vee (A < 18 \wedge A \geq 20) \vee (A < 18 \wedge A > 18). \end{aligned}$$

В полученном выражении имеются два тождественно ложных слагаемых:

$$(A < 18 \wedge A \geq 20) \text{ и } (A < 18 \wedge A > 18).$$

Из выражения $(A \leq 20 \wedge A \geq 20)$ следует, что $A = 20$. Но это противоречит первому сомножителю исходного выражения: $(A \equiv 20)$.

Рассмотрим выражение $(A \leq 20 \wedge A > 18)$. Так как $A \neq 20$, то единственное число, которое удовлетворяет приведенному выражению, — это 19.

Ответ: 19.

Задача 8. X, Y, Z — целые числа, для которых истинно высказывание: $((Z < X) \vee (Z < Y)) \wedge \neg((Z + 1) < X) \wedge \neg((Z + 1) < Y)$. Чему равно Z , если $X = 10, Y = 0$?

Решение.

Упростим исходное высказывание (получим сомножители без отрицания):

$$\begin{aligned} & ((Z < X) \vee (Z < Y)) \wedge \neg((Z + 1) < X) \wedge \neg((Z + 1) < Y) = \\ & = ((Z < X) \vee (Z < Y)) \wedge ((Z + 1) \geq X) \wedge ((Z + 1) \geq Y). \end{aligned}$$

Подставим в выражение значения X и Y :

$$\begin{aligned} & ((Z < 10) \vee (Z < 0)) \wedge ((Z + 1) \geq 10) \wedge ((Z + 1) \geq 0) = \\ & = ((Z < 10) \vee (Z < 0)) \wedge ((Z \geq 9) \wedge (Z \geq -1)). \end{aligned}$$

Рассмотрим выражение:

$$((Z \geq 9) \wedge (Z \geq -1)).$$

Заменим его на выражение $Z \geq 9$, так как если $Z \geq 9$ истинно, то истинно будет и выражение $Z \geq -1$, а при ложности $Z \geq 9$ ложно будет и выражение $Z \geq -1$.

Следовательно:

$$((Z < 10) \vee (Z < 0)) \wedge (Z \geq 9) \wedge (Z \geq -1) = ((Z < 10) \vee (Z < 0)) \wedge (Z \geq 9).$$

Раскроем скобки:

$$((Z < 10) \vee (Z < 0)) \wedge (Z \geq 9) = (Z < 10) \wedge (Z \geq 9) \vee (Z < 0) \wedge (Z \geq 9).$$

Так как $(Z < 0) \wedge (Z \geq 9)$ — тождественно ложное выражение, то:

$$(Z < 10) \wedge (Z \geq 9) \vee (Z < 0) \wedge (Z \geq 9) = (Z < 10) \wedge (Z \geq 9).$$

Из полученного выражения следует, что $Z = 9$.

Ответ: 9.

Задача 9. Классный руководитель пожаловался директору, что у него в классе появилась компания из трех человек, один из которых всегда говорит правду, другой всегда лжет, а третий говорит через раз то ложь, то правду. Директор знает, что их зовут Коля, Саша и Миша, но не знает, кто из них правдив, а кто — нет. Однажды все трое прогуляли урок астрономии. Директор знает, что никогда раньше никто из них не прогуливал астрономию. Он вызвал всех троих в кабинет и поговорил с мальчиками. Коля сказал: «Я всегда прогуливаю астрономию. Не верьте тому, что скажет Саша». Саша сказал: «Это был мой первый прогул этого предмета». Миша сказал: «Все, что говорит Коля — правда». Директор понял, кто из них кто. Определите, кто всегда говорит ложь, кто всегда говорит правду, а кто через раз — то правду, то ложь [6].

Решение.

Рассмотрим высказывания каждого ученика, учитывая, что никто из них раньше не прогуливал астрономию.

В первой части своего высказывания Коля утверждает, что он **всегда** прогуливает астрономию. Это не соответствует истине, так как раньше ученики астрономию не прогуливали. Саша же утверждает, что он прогулял астрономию в первый раз, и данное утверждение не противоречит имеющейся информации. Следовательно, Коля во второй части своего утверждения тоже лжет, так как советует директору не верить Саше. Таким образом, Коля лжет всегда.

Миша в своем высказывании утверждает, что Коля говорит правду. Однако только что было выяснено, что Коля всегда лжет. Следовательно, Миша тоже сказал ложь.

Таким образом, Миша говорит правду через раз, Коля всегда лжет, а Саша всегда говорит правду.

Ответ: Коля всегда лжет, Саша всегда говорит правду, а Миша говорит правду через раз.

Задача 10. Восемь школьников, оставшихся в кабинете на перемене, были вызваны к директору. Один из них разбил окно в кабинете. На вопрос директора кто это сделал, были получены следующие ответы:

- 1) Соня: «Это сделал Володя»;
- 2) Миша: «Это ложь!»;
- 3) Володя: «Я разбил!»;
- 4) Аня: «Это я разбила!»;
- 5) Оля: «Аня не разбивала!»;
- 6) Рома: «Разбила Соня»;
- 7) Коля: «Девочки этого не делали»;
- 8) Толя: «Коля разбил!».

Кто разбил окно, если известно, что из этих восьми высказываний истинны только три?

Решение.

Введем обозначения для ответов школьников, используя первые буквы имени школьников при записи утверждений:

- 1) Соня: «Это сделал Володя» = B ;
- 2) Миша: «Это ложь!» = \bar{B} ;
- 3) Володя: «Я разбил!» = B ;
- 4) Аня: «Это я разбила!» = A ;
- 5) Оля: «Аня не разбивала!» = \bar{A} ;
- 6) Рома: «Разбила Соня» = C ;

7) Коля: «Девочки этого не делали» = $\overline{O} \wedge \overline{A} \wedge \overline{C}$;

8) Толя: «Коля разбил!» = K .

Оформим решение задачи в виде таблицы. В первом столбце этой таблицы укажем имена школьников, которые, как предполагается, могли разбить окно, т. е. школьников, которые хотя бы один раз были упомянуты в высказываниях. В заголовке таблицы указаны обозначения ответов школьников. На пересечении каждой строки и столбца находится логическая величина, указывающая истинность или ложность утверждения, находящегося в заголовке столбца при условии, что окно разбил школьник, имя которого указано в начале строки. В последней колонке таблицы указано количество истинных высказываний (число единиц в строке).

	B	\overline{B}	V	A	\overline{A}	C	$\overline{O} \wedge \overline{A} \wedge \overline{C}$	K	Истинно
Володя	1	0	1	0	1	0	1	0	4
Аня	0	1	0	1	0	0	0	0	2
Соня	0	1	0	0	1	1	0	0	3
Оля	0	1	0	0	1	0	0	0	2
Коля	0	1	0	0	1	0	1	1	4

Рассмотрим первую строку таблицы. Допустим, что окно разбил Володя, тогда:

- 1) утверждение Сони — истинно: $\overline{B} = 1$ (окно разбил Володя);
- 2) утверждение Миши — ложно: $\overline{B} = 0$ (утверждение о том, что окно Володя не разбивал, — ложно);
- 3) утверждение Володи — истинно: $B = 1$;
- 4) утверждение Ани — ложно: $A = 0$ (окно разбил Володя, следовательно, не Аня);
- 5) утверждение Оли — истинно: $\overline{A} = 1$ (Аня окно не разбивала);
- 6) утверждение Ромы — ложно: $C = 0$ (Соня окно не разбивала);
- 7) утверждение Коли — истинно: $\overline{O} \wedge \overline{A} \wedge \overline{C} = 1$ (ни одна из девочек окно не разбивала, так как его разбил Володя);
- 8) утверждение Толи — ложно: $K = 0$ (Коля окно не разбивал).

На основе подобных рассуждений сформированы и остальные строки таблицы. Таким образом, количество истинных выражений становится равным 3 при условии истинности гипотезы «Разбила Соня».

Ответ: окно разбила Соня.

Задача 11. Восемь школьников, оставшихся в кабинете на перемене, были вызваны к директору. Один из них разбил окно в кабинете. На вопрос директора, кто это сделал, были получены следующие ответы:

- 1) Егор: «Разбил Андрей!»;
- 2) Света: «Вика разбила!»;
- 3) Оля: «Разбила Света!»;
- 4) Миша: «Это кто-то с улицы!»;
- 5) Надя: «Да, Оля права ...»;
- 6) Коля: «Это либо Вика, либо Света!»;
- 7) Андрей: «Ни Вика, ни Света этого не делали»;
- 8) Вика: «Андрей не бил».

Кто разбил окно, если известно, что из этих высказываний истинны только три?

Решение.

Введем обозначения для ответов:

- 1) Егор: «Разбил Андрей!» = A ;
- 2) Света: «Вика разбила!» = B ;
- 3) Оля: «Разбила Света!» = C ;
- 4) Миша: «Это кто-то с улицы!» = $\bar{E} \wedge \bar{C} \wedge \bar{B} \wedge \bar{O} \wedge \bar{H} \wedge \bar{K} \wedge \bar{A} \wedge \bar{M}$;
- 5) Надя: «Да, Оля права...» = C ;
- 6) Коля: «Это либо Вика, либо Света!» = $B \oplus C$;
- 7) Андрей: «Ни Вика, ни Света этого не делали» = $\bar{B} \wedge \bar{C}$;
- 8) Вика: «Андрей не бил» = A .

Запишем решение задачи в виде таблицы.

	A	B	C	$\bar{E} \wedge \bar{C} \wedge \bar{B}$	C	$B \oplus C$	$\bar{B} \wedge \bar{C}$	\bar{A}	Истинно
$A = 1$	1	0	0	0	0	0	1	0	2
$B = 1$	0	1	0	0	0	1	0	1	3
$C = 1$	0	0	1	0	1	1	0	1	4

Из таблицы видно, что количество истинных выражений равно 3 при условии истинности гипотезы «Разбила Вика».

Ответ: окно разбила Вика.

Задача 12. Три свидетеля дорожного происшествия сообщили сведения об автомобиле скрывшегося нарушителя. Иван утверждал, что он был на белом «Рено», Сергей сказал, что нарушитель уехал на синей «Тойоте», а Николай сказал, что машина нарушителя была не белая, и, по всей видимости, это был «Форд».

Когда была найдена машина, выяснилось, что каждый свидетель точно определил только один из параметров автомобиля, а в другом ошибся. Определите марку и цвет машины нарушителя.

Решение.

1-й способ.

Введем обозначения для высказываний свидетелей:

- 1) W — машина белого цвета;
- 2) B — машина синего цвета;
- 3) R — марка автомобиля — «Рено»;
- 4) T — марка автомобиля «Тойота»;
- 5) F — марка автомобиля «Форд».

Учитывая, что каждый из свидетелей ошибся только один раз, т. е. хотя бы одно из его высказываний истинно, запишем исходные суждения следующим образом:

- 1) Иван: $W \vee R = 1$;
- 2) Сергей: $B \vee T = 1$;
- 3) Николай: $\overline{W} \vee F = 1$.

Перемножим высказывания свидетелей, зная, что дизъюнкция высказываний равна 1:

$$(W \vee R) \wedge (B \vee T) \wedge (\overline{W} \vee F) = 1.$$

Перемножим выражения, находящиеся в первых двух скобках:

$$\begin{aligned} & (W \vee R) \wedge (B \vee T) \wedge (\overline{W} \vee F) = \\ & = ((W \wedge B) \vee (R \wedge B) \vee (T \wedge W) \vee (T \wedge R)) \wedge (\overline{W} \vee F). \end{aligned}$$

Упростим полученное выражение. Произведение $(W \wedge B)$ является ложным тождеством, так как автомобиль не может быть одновременно и синего, и белого цвета. Выражение $(T \wedge R)$ также ложно, так как марка у автомобиля может быть только одна.

В результате преобразований мы получим следующее выражение:

$$\begin{aligned} & ((W \wedge B) \vee (R \wedge B) \vee (T \wedge W) \vee (T \wedge R)) \wedge (\overline{W} \vee F) = \\ & = ((R \wedge B) \vee (T \wedge W)) \wedge (\overline{W} \vee F) = \\ & = (R \wedge B \wedge \overline{W}) \vee (T \wedge W \wedge \overline{W}) \vee (R \wedge B \wedge F) \vee (T \wedge W \wedge F). \end{aligned}$$

Рассмотрим последнее выражение. Высказывание $(R \wedge B \wedge \overline{W})$ — истинно (марка автомобиля нарушителя — «Рено», цвет автомобиля — синий (не белый)).

Высказывание $(T \wedge W \wedge \overline{W})$ — ложно, так как цвет автомобиля одновременно не может быть и белым, и не белым.

Высказывание $(R \wedge B \wedge F)$ — ложно, так как автомобиль не может относиться одновременно к двум маркам.

Высказывание $(T \wedge W \wedge F)$ ложно по той же причине.

Таким образом,

$$\begin{aligned} & (R \wedge B \wedge \overline{W}) \vee (T \wedge W \wedge \overline{W}) \vee (R \wedge B \wedge F) \vee (T \wedge W \wedge F) = \\ & = (R \wedge B \wedge \overline{W}) = 1. \end{aligned}$$

Следовательно, марка автомобиля — «Рено», а цвет автомобиля — синий (не белый). То есть нарушитель скрылся на синем «Рено».

2-й способ.

Решим задачу методом рассуждений. Допустим, что Иван не ошибся в цвете автомобиля, но ошибся в его марке. Следовательно, машина белая и не «Рено». Тогда Сергей ошибся в цвете и верно сообщил марку. Но тогда из высказываний двух этих очевидцев следует, что нарушитель скрылся на белой «Тойоте», а значит, Николай ошибся и в цвете автомобиля, и в его марке, — получено противоречие с условиями задачи.

Повторно рассмотрим исходное условие, изменив гипотезу о нем. Допустим, что Иван ошибся в цвете автомобиля и не ошибся в марке (машина не белая и «Рено»), а Сергей ошибся в марке автомобиля и не ошибся в цвете (синий «Рено»). Полученное утверждение не противоречит высказыванию Ивана (который не ошибся в цвете и ошибся в марке автомобиля).

Ответ: синий «Рено».

Задача 13. Перед началом футбольного чемпионата Европы болельщики высказали следующие предположения по поводу своих кумиров:

1) Иван: победит сборная Англии, а сборная Франции получит серебряную медаль;

2) Сергей: сборная Франции будет третьей, сборная России будет первой;

3) Николай: англичане проиграют, выиграет сборная Германии.

Когда соревнования закончились, оказалось, что каждый из болельщиков был прав только в одном из своих прогнозов. Какие места на турнире заняли сборные России, Англии, Франции и Германии?

Решение.

1-й способ.

Запишем высказывания болельщиков в виде таблицы:

	Иван	Сергей	Николай
1	Англия	Россия	Германия
2	Франция		
3		Франция	
4			Англия

При решении задачи необходимо учитывать, что две команды не могут занимать одно место.

Допустим, что англичане заняли первое место. Тогда Сергей ошибся, поставив на первое место сборную России. Ошибся и Николай, поставив на первое место сборную Германии. Было выдвинуто предположение, что англичане заняли первое место, но в соответствии с высказыванием Николая получается, что сборная Англии занимает четвертое место. Следовательно, утверждение Ивана о том, что сборная Англии занимает первое место, — ложно. В результате мы получим следующую таблицу (курсивом в ней выделены ложные утверждения, а жирным шрифтом — истинные):

	Иван	Сергей	Николай
1	<i>Англия</i>	Россия	Германия
2	Франция		
3		Франция	
4			Англия

Предположим теперь, что Иван ошибся в первом утверждении и сборная Франции находится на втором месте. Рассматривая высказывания Сергея, заключим, что он ошибся, поставив сборную Франции на третье место.

	Иван	Сергей	Николай
1	<i>Англия</i>	Россия	Германия
2	Франция		
3		<i>Франция</i>	
4			Англия

Следовательно, высказывание Сергея о том, что сборная России находится на первом месте, — истинно. Высказывание же Николая о том, что сборная Германии находится на первом месте, — ложно. Следовательно, сборная Англии находится на четвертом месте.

	Иван	Сергей	Николай
1	<i>Англия</i>	Россия	<i>Германия</i>
2	Франция		
3		<i>Франция</i>	
4			Англия

Таким образом, мы получаем, что сборная России находится на первом месте, сборная Франции — на втором, сборная Англии — на четвертом, а сборная Германии — на третьем месте.

2-й способ.

Введем следующие обозначения:

- 1) A_1 – сборная Англии на первом месте;
- 2) A_4 – сборная Англии на четвертом месте;
- 3) G_1 – сборная Германии на первом месте;
- 4) R_1 – сборная России на первом месте;
- 5) F_2 – сборная Франции на втором месте;
- 6) F_3 – сборная Франции на третьем месте.

Запишем высказывания болельщиков, используя введенные обозначения:

- 1) Иван: $A_1 \vee F_2 = 1$;
- 2) Сергей: $R_1 \vee F_3 = 1$;
- 3) Николай: $G_1 \vee A_4 = 1$.

Запись типа $G_1 \vee A_4 = 1$ означает, что одно из высказываний истинно.

Зная, что каждый болельщик ошибся только один раз, запишем:

$$(A_1 \vee F_2) \wedge (R_1 \vee F_3) \wedge (G_1 \vee A_4) = 1.$$

Преобразуем полученное высказывание (перемножим выражения в первых двух скобках):

$$((A_1 \wedge R_1) \vee (A_1 \wedge F_3) \vee (F_2 \wedge R_1) \vee (F_2 \wedge F_3)) \wedge (R_1 \vee F_3) \wedge (G_1 \vee A_4).$$

Высказывание $(A_1 \wedge R_1)$ — ложно, так как две сборные одновременно не могут находиться на первом месте. Высказывание $(F_2 \wedge F_3)$ также ложно, так как сборная Франции не может одновременно находиться на 2-м и 3-м местах. Следовательно, получим:

$$((A_1 \wedge F_3) \vee (F_2 \wedge R_1)) \wedge (R_1 \vee F_3) \wedge (G_1 \vee A_4).$$

Раскроем скобки (применяя распределительный закон):

$$\begin{aligned} & ((A_1 \wedge F_3) \vee (F_2 \wedge R_1)) \wedge (R_1 \vee F_3) \wedge (G_1 \vee A_4) = \\ & = (A_1 \wedge F_3 \wedge G_1) \vee (A_1 \wedge F_3 \wedge A_4) \vee (F_2 \wedge R_1 \wedge G_1) \vee (F_2 \wedge R_1 \wedge A_4). \end{aligned}$$

Высказывания $(A_1 \wedge F_3 \wedge G_1)$ и $(F_2 \wedge R_1 \wedge G_1)$ тождественно ложны, так как одновременно две сборные не могут занимать первое место. Высказывание $(A_1 \wedge F_3 \wedge A_4)$ также ложно, так как сборная Англии не может одновременно находиться на 1-м и 4-м местах. Получаем:

$$\begin{aligned} & (A_1 \wedge F_3 \wedge G_1) \vee (A_1 \wedge F_3 \wedge A_4) \vee (F_2 \wedge R_1 \wedge G_1) \vee (F_2 \wedge R_1 \wedge A_4) = \\ & = (F_2 \wedge R_1 \wedge A_4). \end{aligned}$$

Из последнего высказывания следует, что сборная России находится на первом месте, сборная Франции на втором, а сборная Англии — на четвертом. Следовательно, сборная Германии занимает третье место.

Ответ: сборная России — первое место, сборная Франции — второе, сборная Германии — третье, сборная Англии — четвертое.

Задача 14. Мама, прибежавшая на звон разбившейся вазы, застала всех трех своих сыновей в совершенно невинных позах: Саша, Ваня и Коля делали вид, что происшедшее к ним не относится. Однако футбольный мяч среди осколков явно говорил об обратном.

— Кто это сделал? — спросила мама.

— Коля не бил по мячу, — сказал Саша. — Это сделал Ваня.

Ваня ответил:

— Разбил Коля, Саша не играл в футбол дома.

— Так я и знала, что вы друг на дружку сваливать будете, рассердилась мама. Ну, а ты что скажешь? — спросила она Колю.

— Не сердись, мамочка! Я знаю, что Ваня не мог этого сделать. А я сегодня еще не сделал уроки, — сказал Коля.

Оказалось, что один из мальчиков оба раза солгал, а двое в каждом из своих заявлений говорили правду. Кто разбил вазу [2]?

Решение.

Введем обозначения для высказываний:

1) C : вазу разбил Саша;

2) B : вазу разбил Ваня;

3) K : вазу разбил Коля.

Запишем высказывания с помощью введенных обозначений:

1) Саша: \bar{K}, \bar{B} ;

2) Ваня: \bar{K}, \bar{C} ;

3) Коля: B .

(Отметим, что высказывание Коли о том, что он не сделал уроки, не несет информации относительно того, кто разбил вазу.)

В тексте задачи сказано, что один из мальчиков оба раза солгал, а двое в каждом из своих заявлений говорили правду.

Рассмотрим случай, когда Коля солгал (\bar{B}), а Саша и Ваня сказали правду ($\bar{K} \wedge B$ и $K \wedge \bar{C}$). Запишем следующее выражение:

$$(\bar{B}) \wedge (\bar{K} \wedge B) \wedge (K \wedge \bar{C}) = B \wedge \bar{K} \wedge B \wedge K \wedge \bar{C}.$$

Данное выражение ложно, так как $\bar{K} \wedge K = 0$.

Рассмотрим теперь ситуацию, когда Ваня солгал ($\bar{K} \wedge \bar{C}$), а Коля и Саша сказали правду (B и $\bar{K} \wedge B$). Запишем логическое выражение, описывающее данную ситуацию:

$$\bar{K} \wedge \bar{C} \wedge \bar{K} \wedge B \wedge \bar{B} = \bar{K} \wedge \bar{C} \wedge B \wedge \bar{B}.$$

Данное выражение ложно, так как $\bar{B} \wedge B = 0$.

Допустим, что Саша два раза солгал и, следовательно, оба его утверждения неверны:

$$\bar{\bar{K}} \wedge \bar{\bar{B}} = K \wedge B.$$

Если Саша солгал, то Ваня и Коля сказали правду:

$$K \wedge \bar{C} \wedge \bar{B}.$$

Рассматривая ситуацию как истинную, запишем следующее выражение:

$$K \wedge \bar{B} \wedge K \wedge \bar{C} \wedge \bar{B} = K \wedge \bar{B} \wedge \bar{C}.$$

Данное выражение истинно. Следовательно, вазу разбил Коля.

Ответ: Коля.

Задача 15. В пяти домах разного цвета живут люди пяти национальностей. Они курят пять разных сортов сигарет, пьют пять разных напитков и содержат пять разных видов животных.

- 1) англичанин живет в красном доме;
- 2) швед держит собак;
- 3) датчанин пьет чай;
- 4) зеленый дом слева от белого;
- 5) хозяин зеленого дома любит пить кофе;
- 6) любитель Pall Mall содержит птиц;
- 7) хозяин желтого дома курит Dunhill;
- 8) в центральном доме предпочитают молоко;
- 9) норвежец живет в первом доме;
- 10) курящий Blend живет по соседству с хозяином кошек;
- 11) хозяин лошадей живет рядом с тем, кто курит Dunhill;
- 12) любитель пива курит Bluemasters;
- 13) немец курит Prince;
- 14) норвежец живет рядом с синим домом;
- 15) курящий Blend живет по соседству с пьющим воду.

Определите национальность человека, содержащего рыбок.

Решение.

1-й способ.

Решение удобно оформить в виде таблицы:

	Дом 1 (лев.)	Дом 2	Дом 3	Дом 4	Дом 5 (прав.)
Национальность					
Цвет дома					
Напиток					
Сигареты					
Животное					

1. Занесем в эту таблицу факты, которые не вызывают сомнения (в скобках указан номер соответствующего высказывания из условия задачи):

- 1.1) норвежец живет в первом доме (9);

- 1.2) норвежец живет рядом с синим домом (14);
- 1.3) в центральном доме предпочитают молоко (8).
2. Определим цвет дома норвежца:
 - 2.1) цвет дома — не синий, так как синий — это второй дом;
 - 2.2) цвет дома не может быть красным, так как в красном живет англичанин (1);
 - 2.3) цвет дома не может быть зеленым или белым, так как зеленый дом находится слева от белого (т. е. они стоят рядом друг с другом), а рядом с первым домом находится синий дом;
 - 2.4) следовательно, дом норвежца — желтый.
3. Хозяин желтого дома курит Dunhill (7).
4. Определим напиток норвежца:
 - 4.1) норвежец не пьет чай, так как чай пьет датчанин (3);
 - 4.2) норвежец не пьет кофе, так как его пьют в зеленом доме (5);
 - 4.3) норвежец не пьет пиво, так как пиво пьет любитель Bluemasters (12), а норвежец курит Dunhill;
 - 4.4) норвежец не пьет молоко, так как его пьют в центральном доме (8);
 - 4.5) следовательно, норвежец пьет воду.
5. Курящий Blend живет по соседству с пьющим воду (15). Относительно человека, курящего Blend, есть еще одно высказывание: «Курящий Blend живет по соседству с хозяином кошек» (10). Однако неизвестно, слева или справа живет сосед, который держит кошек. Поэтому данное утверждение будет рассмотрено чуть позже.
6. Хозяин лошадей живет рядом с тем, кто курит Dunhill (11).
7. Расставим цвета остальных домов:
 - 7.1) зеленый и белый дома стоят рядом (4) — это могут быть дома 3 и 4 или 4 и 5;
 - 7.2) хозяин зеленого дома пьет кофе (5), а в центральном доме пьют молоко (8), следовательно, зеленый — это дом под номером 4, а белый (стоящий рядом) — дом под номером 5;
 - 7.3) получаем, что центральный дом — красный.
8. В зеленом доме пьют кофе (5).
9. Англичанин живет в красном доме (1).
10. Рассмотрим высказывание: «Любитель пива курит Bluemasters» (12). В первом, третьем и четвертом домах уже пьют некоторый напиток; кроме того, во втором доме курят Blend. Следовательно, любитель пива живет в доме 5.
11. Датчанин пьет чай. Единственный дом, для которого еще не найден напиток, — 2. Следовательно, датчанин живет во втором доме и пьет чай.

12. Определим дом, в котором живет немец. Немец курит Prince (13). Национальности людей, живущих в первом, втором и третьем домах, нам уже известны, а в пятом доме курят Bluemasters. Следовательно, немец живет в доме 4.

13. Любитель Pall Mall держит птиц (6). Единственный дом, для которого не определены сигареты, — дом 3. Следовательно, в доме 3 курят Pall Mall и держат птиц.

14. Швед держит собак (2). Получаем, что швед живет в доме 5.

15. Курящий Blend живет по соседству с хозяином кошек (10). Получаем, что животное в первом доме — кошки.

16. Из полученной таблицы можно заключить, что в доме 4 держат рыбок, а их владелец — немец.

	Дом 1 (лев.)	Дом 2	Дом 3	Дом 4	Дом 5 (прав.)
Национальность	Норвежец (1.1)	Датчанин (10)	Англичанин (3)	Немец (11)	Швед (13)
Цвет дома	Желтый (4)	Синий (1.2)	Красный (3)	Зеленый (2.3)	Белый (2.3)
Напиток	Вода (8.3)	Чай (10)	Молоко (1.3)	Кофе (7)	Пиво (9)
Сигареты	Dunhill (5)	Blend (8.3)	Pall Mall (12)	Prince (11)	Bluemasters (9)
Животное	Кошки (14)	Лошадь (6)	Птицы (12)	Рыбки (15)	Собаки (13)

Примечание. Здесь и далее числа, приведенные в скобках, указывают номера пунктов решения, в которых делается вывод о том, что нужно поместить в рассматриваемую ячейку таблицы.

2-й способ.

1. Занесем в таблицу факты, которые не вызывают сомнения:

1.1) норвежец живет в первом доме (9);

1.2) норвежец живет рядом с синим домом (14);

1.3) в центральном доме предпочитают молоко (8).

2. Определим номера цвета домов:

2.1) зеленый и белый дома должны находиться рядом (4), следовательно, это не могут быть дома 1 и 2 ;

2.2) допустим, что зеленый дом — под номером 3. Зеленый дом находится слева от белого, т. е. сначала идет зеленый дом, а потом — белый. Из (5) следует, что в зеленом доме пьют кофе, следовательно,

зеленый дом не может быть под номером 3, так как в третьем доме пьют молоко;

2.3) получаем, что зеленый дом — под номером 4, а белый — под номером 5.

3. Англичанин живет в красном доме (1), следовательно, он не может жить в первом доме (так как в нем живет норвежец), а также в домах 2, 4 и 5 (так как для них определен цвет). Получаем, что англичанин живет в доме 3.

4. Из приведенных выше утверждений получаем, что норвежец живет в желтом доме.

5. Хозяин желтого дома курит Dunhill (7).

6. Хозяин лошадей живет рядом с тем, кто курит Dunhill (11).

7. В зеленом доме пьют кофе (5).

8. Определим, где живет курящий Blend, используя высказывание 15. На данном этапе решения задачи мы получаем следующую таблицу:

	Дом 1 (лев.)	Дом 2	Дом 3	Дом 4	Дом 5 (прав.)
Национальность	Норвежец (1.1)		Англи- чанин (3)		
Цвет дома	Желтый (4)	Синий (1.2)	Красный (3)	Зеленый (2.3)	Белый (2.3)
Напиток			Молоко (1.3)	Кофе (7)	
Сигареты	Dunhill (5)				
Животное		Лошадь (6)			

Курящий Blend не может жить в доме 1 (так как в первом доме курят Dunhill) и в доме 4 (так как в доме, расположенном рядом с домом, в котором курят Blend, пьют воду). Следовательно, курить Blend могут в домах 2, 3 или 4. Тогда:

8.1) допустим, что Blend курят в доме 4. Получаем, что в доме 5 пьют воду:

	Дом 1 (лев.)	Дом 2	Дом 3	Дом 4	Дом 5 (прав.)
Национальность	Норвежец (1.1)	Датча- нин	Англи- чанин (3)		
Цвет дома	Желтый (4)	Синий (1.2)	Красный (3)	Зеленый (2.3)	Белый (2.3)
Напиток		Чай	Молоко (1.3)	Кофе (7)	Вода
Сигареты	Dunhill (5)			Blend	
Животное		Лошадь (6)			

Из утверждения 3 получаем, что во втором доме живет датчанин и пьет чай. Но тогда полученная таблица противоречит утверждению 12, так как в данной таблице нельзя определить, в каком доме пьют пиво и курят Bluemasters. Следовательно, Blend курят не в доме 4;

8.2) допустим, что Blend курят в доме 3. Получаем, что в доме 2 пьют воду.

	Дом 1 (лев.)	Дом 2	Дом 3	Дом 4	Дом 5 (прав.)
Национальность	Норвежец (1.1)		Англи- чанин (3)		Датчанин
Цвет дома	Желтый (4)	Синий (1.2)	Красный (3)	Зеленый (2.3)	Белый (2.3)
Напиток		Вода	Молоко (1.3)	Кофе (7)	Чай
Сигареты	Dunhill (5)		Blend		
Животное		Лошадь (6)			

Из утверждения 3 получаем, что в пятом доме живет датчанин и пьет чай, но тогда полученная таблица противоречит утверждению 12, так как в данной таблице нельзя определить, в каком доме пьют пиво и курят Bluemasters. Следовательно, Blend курят не в доме 3;

8.3) допустим, что Blend курят в доме 2. Получаем, что в доме 1 пьют воду.

9. Рассмотрим выражение: «Любитель пива курит Bluemasters». Но в первом, третьем и четвертом домах уже пьют некоторый напиток; также во втором доме курят Blend. Следовательно, любитель пива живет в доме 5.

10. Датчанин пьет чай. Единственный дом, для которого еще не найден напиток, — 2. Следовательно, датчанин живет во втором доме и пьет чай.

11. Определим, в каком доме живет немец. Немец курит Prince (13). Национальности людей, живущих в первом, втором и третьем домах, нам уже известны, а в пятом доме курят Bluemasters. Следовательно, немец живет в 4 доме.

12. Любитель Pall Mall держит птиц (6). Единственный дом, для которого еще не определены сигареты, — дом 3. Следовательно, в доме 3 курят Pall Mall и держат птиц.

13. Швед держит собак (2). Получаем, что швед живет в доме 5.

14. Курящий Blend живет по соседству с хозяином кошек (10). Получаем, что животное из первого дома — кошки.

15. Из полученной таблицы можно заключить, что в доме 4 держат рыбок, а их владелец — немец.

Ответ: Рыбок держит немец.

Задача 16. На одной улице стоят в ряд 4 дома, в которых живут 4 человека: Алексей, Егор, Виктор и Михаил. Известно, что каждый из них владеет ровно одной из следующих профессий: Токарь, Столяр, Хирург и Окулист, но неизвестно, кто какой, и неизвестно, кто в каком доме живет. Однако известно, что:

- 1) Токарь живет левее Столяра;
- 2) Хирург живет правее Окулиста;
- 3) Окулист живет рядом со Столяром;
- 4) Токарь живет не рядом со Столяром;
- 5) Виктор живет правее Окулиста;
- 6) Михаил не Токарь;
- 7) Егор живет рядом со Столяром;
- 8) Виктор живет левее Егора;

Выясните, кто какой профессии, и кто где живет [7].

Решение.

Из утверждения 4 следует, что между Токарем и Столяром кто-то живет. Но это не может быть Хирург, так как он живет правее Столяра (утверждение 2), а Токарь живет левее Столяра (утверждение 1). Поэтому между Токарем и Столяром живет Окулист (утверждение 3).

Получаем следующее распределение:

| Токарь, Окулист, Столяр, Хирург |.

Далее, Виктор живет правее Окулиста (утверждение 5). Следовательно, Виктор может быть Столяром или Хирургом. Так как

Виктор живет левее Егора (утверждение 8), а Егор живет рядом со Столяром (утверждение 7), то Егор — Хирург, а Виктор — Столяр. Михаил не является Токарем (утверждение 6), следовательно, он — Окулист. Получаем, что Алексей — Токарь.

В итоге получаем следующее распределение: | Алексей, Михаил, Виктор, Егор |.

Ответ: Алексей (Токарь), Михаил (Окулист), Виктор (Столяр), Егор (Хирург).

2.3. Построение таблиц истинности логических выражений

Таблица истинности логической формулы показывает соответствие между возможными наборами значений переменных и значениями формулы, которые она принимает при указанных значениях переменных.

При записи таблицы истинности логической формулы важно безошибочно перечислить *все* сочетания значений переменных, входящих в формулу.

Допустим, что в формуле участвуют три логические переменные: A , B и C . Тогда возможно 8 ($N = 2^3$) сочетаний их значений.

Для перечисления всех таких сочетаний можно использовать следующий способ.

1. Записываем значения переменной C . Список значений этой переменной представим в виде четырех групп, каждая из которых состоит из одного нуля и одной единицы: 01 01 01 01.

2. Записываем значения переменной B . Этот список значений представим в виде двух групп, в каждой из которых находятся два нуля и две единицы: 0011 0011.

3. Записываем значения переменной A . Этот список представим в виде одной группы, в которой находятся четыре нуля и четыре единицы: 00001111.

В результате мы получим таблицу, в которой перечислены все возможные сочетания значений переменных.

A	B	C
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Пример. Составим таблицу истинности для формулы $(A \wedge B) \vee (A \vee B)$. В первых двух столбцах таблицы запишем четыре возможные пары значений переменных A и B . В последующих столбцах запишем значения промежуточных формул. В последнем столбце запишем значение заданной формулы для каждой пары значений переменных. Получим таблицу:

Переменные		Промежуточные формулы			Результат
A	B	$A \wedge B$	$A \vee B$	$\overline{A \vee B}$	$(A \wedge B) \vee (\overline{A \vee B})$
0	0	0	0	1	1
0	1	0	1	0	0
1	0	0	1	0	0
1	1	1	1	0	1

Задача 1. Определите, является ли эквивалентным выражению $A \wedge \neg(\neg B \vee C)$ высказывание $A \wedge B \wedge \neg C$ [6].

Решение.

1-й способ.

Упростим исходное логическое выражение, используя правило де Моргана и операции переменной с ее инверсией:

$$A \wedge \neg(\neg B \vee C) = A \wedge \neg(\neg B) \wedge \neg C = A \wedge B \wedge \neg C.$$

Получаем, что два указанных в условиях задачи высказывания эквивалентны.

2-й способ.

Составим таблицы истинности данных выражений и сравним их.

A	B	C	\overline{B}	\overline{C}	$A \wedge B \wedge \neg C$	$A \wedge \neg(\neg B \vee C)$
0	0	0	1	1	0	0
0	0	1	1	0	0	0
0	1	0	0	1	0	0
0	1	1	0	0	0	0
1	0	0	1	1	0	0
1	0	1	1	0	0	0
1	1	0	0	1	1	1
1	1	1	0	0	0	0

Судя по полученной таблице истинности, заданные выражения эквивалентны друг другу.

Ответ: выражения эквивалентны.

Задача 2. Определите, соответствует ли логическое выражение $F = \neg X \wedge \neg Y \wedge \neg Z$ приведенному ниже фрагменту таблицы истинности [6].

X	Y	Z	F
1	0	0	1
0	0	0	1
1	1	1	0

Решение.

Подставим значения переменных, находящиеся в каждой строке этой таблицы истинности, в предложенное выражение и вычислим его значение. Выражение соответствует таблице истинности, если для всех строк таблицы истинности полученное значение соответствует величине F в соответствующей строке:

$$1) X = 1, Y = 0, Z = 0: \overline{1} \wedge \overline{0} \wedge \overline{0} = 0 \wedge 1 \wedge 1 = 0, F = 1;$$

$$2) X = 0, Y = 0, Z = 0: \overline{0} \wedge \overline{0} \wedge \overline{0} = 1 \wedge 1 \wedge 1 = 1, F = 1;$$

$$3) X = 1, Y = 1, Z = 1: \overline{1} \wedge \overline{1} \wedge \overline{1} = 0 \wedge 0 \wedge 0 = 0, F = 0.$$

Таким образом, указанное выражение не соответствует таблице истинности выражения F , так как вычисленные значения выражений не соответствуют значениям выражения F .

Примечание! Если в процессе вычислений получено, что для одного из наборов переменных анализируемое выражение не соответствует таблице истинности выражения F , то дальнейшие вычисления можно не выполнять.

Ответ: данное логическое выражение не соответствует приведенному фрагменту таблицы истинности.

Задача 4. Укажите значения логических переменных K, L, M, N , при которых логическое выражение $(\overline{K} \wedge \overline{N}) \rightarrow (M \rightarrow L)$ ложно.

Решение.

При решении данной задачи необходимо учитывать, что исходное выражение будет ложным только для одного набора значений переменных (и это указано в тексте задания).

1-й способ.

Упростим исходное выражение:

$$(\overline{K} \wedge \overline{N}) \rightarrow (M \rightarrow L).$$

Раскроем импликацию (см. формулу (2.1)):

$$(\overline{K} \wedge \overline{N}) \vee (M \rightarrow L).$$

Применим правило де Моргана к выражению $(\overline{K} \wedge \overline{N})$:

$$(\overline{K} \wedge \overline{N}) = \overline{(K \vee N)}.$$

Используя закон двойного отрицания, можно записать:

$$\overline{(\overline{K} \vee \overline{N})} = (K \vee N).$$

Рассмотрим выражение $(M \rightarrow L)$. На основе формулы (2.1) запишем:

$$(M \rightarrow L) = (\overline{M} \vee L).$$

Следовательно,

$$(\overline{K} \wedge \overline{N}) \vee (M \rightarrow L) = K \vee N \vee (\overline{M} \vee L) = K \vee N \vee \overline{M} \vee L.$$

Выражение $K \vee N \vee \overline{M} \vee L$ ложно при $M = 1, L = 0, K = 0, N = 0$, так как $0 \vee 0 \vee 1 \vee 0 = 0$.

Выполним проверку: подставим вычисленные значения переменных в исходное выражение:

$$(0 \wedge 0) \rightarrow (1 \rightarrow 0) = (1 \wedge 1) \rightarrow 0 = 1 \rightarrow 0 = 0.$$

Таким образом, для данных значений переменных выражение ложно.

2-й способ.

Рассмотрим исходное логическое выражение $(\overline{K} \wedge \overline{N}) \rightarrow (M \rightarrow L)$. Данное выражение ложно, если посылка $(\overline{K} \wedge \overline{N})$ истинна, а следствие $(M \rightarrow L)$ — ложно (см. табл. 2.1).

Выражение $(\overline{K} \wedge \overline{N})$ истинно, если логические переменные K и N ложны (см. таблицу истинности для логического умножения).

Выражение $(M \rightarrow L)$ ложно, если значение переменной M будет истинным, а L — ложным.

Следовательно, выражение $(\overline{K} \wedge \overline{N}) \rightarrow (M \rightarrow L)$ ложно при: $M = 1, L = 0, K = 0, N = 0$, так как $(0 \wedge 0) \rightarrow (1 \rightarrow 0) = (1 \wedge 1) \rightarrow 0 = 1 \rightarrow 0 = 0$.

3-й способ.

Построим таблицу истинности выражения.

K	L	M	N	\overline{K}	\overline{N}	$\overline{K} \wedge \overline{N}$ (1)	$M \rightarrow L$ (2)	$(1) \rightarrow (2)$
0	0	0	0	1	1	1	1	1
0	0	0	1	1	0	0	1	1
0	0	1	0	1	1	1	0	0
0	0	1	1	1	0	0	0	1
0	1	0	0	1	1	1	1	1
0	1	0	1	1	0	0	1	1
0	1	1	0	1	1	1	1	1
0	1	1	1	1	0	0	1	1
1	0	0	0	0	1	0	1	1
1	0	0	1	0	0	0	1	1
1	0	1	0	0	1	0	0	1
1	0	1	1	0	0	0	0	1
1	1	0	0	0	1	0	1	1
1	1	0	1	0	0	0	1	1
1	1	1	0	0	1	0	1	1
1	1	1	1	0	0	0	1	1

Из таблицы истинности видно, что исходное выражение ложно только для одного набора переменных: $M = 1, L = 0, K = 0, N = 0$.

Примечание. До построения таблицы истинности можно попробовать упростить исходное выражение, чтобы ускорить построение таблицы. В рассматриваемом случае проще всего построить таблицу истинности для выражения, полученного после упрощения, рассмотренного в первом способе решения, чем для исходного выражения. Однако упростить исходное выражение можно не всегда.

Ответ: $M = 1, L = 0, K = 0, N = 0$.

Задача 5. Сколько различных решений имеет уравнение, где K, L, M и N — логические переменные?

Решение.

1-й способ.

Построим таблицу истинности выражения. (Напомним, что в ряде случаев можно упростить исходное выражение, что позволит ускорить процесс построения таблицы истинности.)

K	L	M	N	$M \vee K$ (1)	$(1) \wedge L$ (2)	$(2) \rightarrow N$
0	0	0	0	0	0	1
0	0	0	1	0	0	1
0	0	1	0	1	0	1
0	0	1	1	1	0	1
0	1	0	0	0	0	1
0	1	0	1	0	0	1
0	1	1	0	1	1	0
0	1	1	1	1	1	1
1	0	0	0	1	0	1
1	0	0	1	1	0	1
1	0	1	0	1	0	1
1	0	1	1	1	0	1
1	1	0	0	1	1	0
1	1	0	1	1	1	1
1	1	1	0	1	1	0
1	1	1	1	1	1	1

В приведенной выше таблице анализируемое выражение истинно для 13 наборов значений переменных.

2-й способ.

Упростим исходное выражение:

1) на основе формулы (2.1) запишем выражение без импликации:

$$((M \vee K) \wedge L) \rightarrow N = \overline{(M \vee K) \wedge L} \vee N;$$

2) применим закон де Моргана:

$$(\overline{M \vee K}) \wedge L \vee N = (\overline{M \vee K}) \vee \overline{L} \vee N = (\overline{M} \vee \overline{K}) \vee \overline{L} \vee N.$$

Данное выражение истинно в 7 случаях:

- 1) $(\overline{M} \wedge \overline{K})$ — ложно, \overline{L} — ложно, N — истинно;
- 2) $(\overline{M} \wedge \overline{K})$ — ложно, \overline{L} — истинно, N — ложно;
- 3) $(\overline{M} \wedge \overline{K})$ — ложно, \overline{L} — истинно, N — истинно;
- 4) $(\overline{M} \wedge \overline{K})$ — истинно, \overline{L} — ложно, N — истинно;
- 5) $(\overline{M} \wedge \overline{K})$ — истинно, \overline{L} — ложно, N — ложно;
- 6) $(\overline{M} \wedge \overline{K})$ — истинно, \overline{L} — истинно, N — истинно;
- 7) $(\overline{M} \wedge \overline{K})$ — истинно, \overline{L} — истинно, N — ложно.

Рассмотрим каждый такой случай отдельно:

1) выражение $(\overline{M} \wedge \overline{K})$ ложно, когда $M = 1, K = 1$ или $M = 0, K = 1$ или $M = 1, K = 0$, т. е. при трех сочетаниях значений переменных, \overline{L} ложно при $L = 1$ ($1 = 0$), а N истинно при $N = 1$. Таким образом, данное выражение истинно для трех наборов значений переменных;

2) выражение $(\overline{M} \wedge \overline{K})$ ложно, когда $M = 1, K = 1$ или $M = 0, K = 1$ или $M = 1, K = 0$, т. е. при трех сочетаниях значений переменных, \overline{L} истинно при $L = 0$, а N ложно при $N = 0$. В данном случае исходное выражение истинно для трех наборов значений переменных;

3) выражение $(\overline{M} \wedge \overline{K})$ ложно, когда $M = 1, K = 1$ или $M = 0, K = 1$ или $M = 1, K = 0$, т. е. при трех сочетаниях значений переменных, \overline{L} истинно при $L = 0$, а N истинно при $N = 1$. В данном случае исходное выражение истинно для трех наборов значений переменных;

4) выражение $(\overline{M} \wedge \overline{K})$ истинно при $M = 0, K = 0$, \overline{L} ложно при $L = 1$, а N истинно при $N = 1$. То есть исходное выражение истинно для одного набора значений переменных;

5) выражение $(\overline{M} \wedge \overline{K})$ истинно при $M = 0, K = 0$, \overline{L} ложно при $L = 1$, а N ложно при $N = 0$. То есть исходное выражение истинно для одного набора значений переменных;

6) выражение $(\overline{M} \wedge \overline{K})$ истинно при $M = 0, K = 0$, \overline{L} истинно при $L = 0$, а N истинно при $N = 1$. В этом случае исходное выражение истинно для одного набора значений переменных;

7) выражение $(\overline{M} \wedge \overline{K})$ истинно при $M = 0, K = 0$, \overline{L} истинно при $L = 0$, а N ложно при $N = 0$. В этом случае исходное выражение истинно для одного набора значений переменных.

Получаем, что исходное выражение истинно в $3 + 3 + 3 + 1 + 1 + 1 + 1 = 13$ случаях.

3-й способ.

Требуется найти количество наборов значений переменных, на которых данное выражение истинно. В выражении участвуют 4 ло-

гические переменные, поэтому существует $2^4 = 16$ возможных сочетаний их значений. Тогда количество наборов переменных (T), на которых выражение истинно, вычисляется следующим образом: $T = 16 - F$, где F — количество наборов переменных, на которых выражение ложно.

Выражение $(\overline{M} \wedge \overline{K}) \vee \overline{L} \vee N$ ложно, когда $(\overline{M} \wedge \overline{K})$ — ложно, \overline{L} — ложно и N — ложно. Выражение $(\overline{M} \wedge \overline{K})$, в свою очередь, ложно, когда $M = 1, K = 1$ или $M = 1, K = 0$ или $M = 0, K = 1$ (т. е. в трех случаях — см. табл. 2.1 истинности для логического умножения), \overline{L} ложно при $L = 1$, а N ложно при $N = 0$.

Таким образом, выражение $(\overline{M} \wedge \overline{K}) \vee \overline{L} \vee N$ ложно в трех случаях; следовательно, истинно оно будет в 13 случаях ($16 - 3 = 13$).

Ответ: 13.

Задача 6. Сколько различных решений имеет уравнение $K \wedge L \wedge M \vee \overline{L} \wedge \overline{M} \wedge N = 1$, где K, L, M и N — логические переменные?

Решение.

1-й способ.

Построим таблицу истинности заданного выражения и найдем, на скольких наборах переменных это выражение будет истинным.

K	L	M	N	$K \wedge L \wedge M$ (1)	$\overline{L} \wedge \overline{M} \wedge N$ (2)	$(1) \vee (2)$
0	0	0	0	0	0	0
0	0	0	1	0	1	1
0	0	1	0	0	0	0
0	0	1	1	0	0	0
0	1	0	0	0	0	0
0	1	0	1	0	0	0
0	1	1	0	0	0	0
0	1	1	1	0	0	0
1	0	0	0	0	0	0
1	0	0	1	0	1	1
1	0	1	0	0	0	0
1	0	1	1	0	0	0
1	1	0	0	0	0	0
1	1	0	1	0	0	0
1	1	1	0	1	0	1
1	1	1	1	1	0	1

В приведенной выше таблице анализируемое выражение истинно для четырех наборов переменных.

2-й способ.

В соответствии с таблицей истинности логического умножения (дизъюнкции) можно сказать, что дизъюнкция истинна, если хотя бы одно из высказываний истинно, т. е.:

1) высказывание $K \wedge L \wedge M$ истинно, а высказывание $\bar{L} \wedge \bar{M} \wedge N$ — ложно;

2) высказывание $K \wedge L \wedge M$ ложно, а высказывание $\bar{L} \wedge \bar{M} \wedge N$ — истинно;

3) и высказывание $K \wedge L \wedge M$ истинно, и высказывание $\bar{L} \wedge \bar{M} \wedge N$ — истинно.

В первом случае высказывание $K \wedge L \wedge M$ истинно, когда $K = L = M = 1$, т. е. только для одного набора значений переменных. Запишем высказывание $\bar{L} \wedge \bar{M} \wedge N$ с учетом найденных значений переменных K, L, M : $\bar{1} \wedge \bar{1} \wedge N = 0 \wedge 0 \wedge N$. Это высказывание ложно при $N = 0$ или $N = 1$, т. е. для двух наборов переменных. Таким образом, в первом случае исходное высказывание будет истинно в $1 \cdot 2 = 2$ вариантах.

Во втором случае высказывание $\bar{L} \wedge \bar{M} \wedge N$ истинно при $L = 0, M = 0, N = 1$, т. е. для одного набора значений. Тогда высказывание $K \wedge L \wedge M$ примет следующий вид: $K \wedge 0 \wedge 0$. Данное выражение ложно при $K = 1$ или $K = 0$, т. е. для двух наборов переменных. Таким образом, исходное высказывание будет истинно в $1 \cdot 2 = 2$ вариантах.

В третьем варианте высказывания $K \wedge L \wedge M$ и $\bar{L} \wedge \bar{M} \wedge N$ одновременно истинными не будут, так как переменные M и L входят в одно из этих выражений с инверсией.

Следовательно, уравнение $K \wedge L \wedge M \wedge \bar{L} \wedge \bar{M} \wedge N = 1$ имеет $2 + 2 = 4$ решения.

Внимание! При анализе выражения выгоднее начать с той части выражения, которая считается истинной, а затем рассмотреть остальные его части.

Ответ: 4.

Задача 7. Определите, является ли тождественно ложной (противоречивой) формула $(A \leftrightarrow C) \wedge \overline{A \rightarrow B} \rightarrow C$.

Решение.

1-й способ.

Тождественно ложная формула принимает значение «ложь» при **любых** значениях входящих в нее логических переменных. Построим таблицу истинности для заданной формулы:

A	B	C	$A \leftrightarrow C$ (1)	$A \rightarrow B$ (2)	$\neg(2)$ (3)	$(3) \rightarrow C$ (4)	$\neg(4)$ (5)	$(1) \wedge (5)$
0	0	0	1	1	0	1	0	0
0	0	1	0	1	0	1	0	0
0	1	0	1	1	0	1	0	0
0	1	1	0	1	0	1	0	0
1	0	0	0	0	1	0	1	0
1	0	1	1	0	1	1	0	0
1	1	0	0	1	0	1	0	0
1	1	1	1	1	0	1	0	0

2-й способ.

Упростим формулу.

$$\begin{aligned}
 (A \leftrightarrow C) \wedge \overline{A \rightarrow B} \rightarrow C &= \{\text{раскрытие равносильности и импликации}\} = \\
 &= (A \wedge C \vee \overline{A} \wedge \overline{C}) \wedge \overline{(A \vee B) \rightarrow C} = (A \wedge C \vee \overline{A} \wedge \overline{C}) \wedge \overline{(A \vee B \vee C)} = \\
 &= (A \wedge C \vee \overline{A} \wedge \overline{C}) \wedge (\overline{A} \vee \overline{B} \vee \overline{C}) = (A \wedge C \vee \overline{A} \wedge \overline{C}) \wedge (\overline{A} \wedge \overline{C} \vee \overline{B}) = \\
 &= A \wedge C \wedge \overline{A} \wedge \overline{C} \vee \overline{A} \wedge \overline{C} \wedge \overline{B} \vee A \wedge \overline{C} \wedge \overline{B} = 0 \vee 0 = 0.
 \end{aligned}$$

Ответ: да, является.

Задача 8. Составьте таблицу истинности для функции $F = \neg((A \leftrightarrow \leftrightarrow B) \rightarrow C) \vee \neg(A \wedge C)$, в которой столбец значений аргумента A представляет собой двоичную запись числа 15, столбец значений аргумента B — запись числа 51, а столбец значений аргумента C — запись числа 85. Числа в столбце записываются сверху вниз от старшего разряда к младшему. Переведите полученную двоичную запись значения функции F в десятичную систему счисления.

Решение.

1-й способ.

Запишем таблицу истинности заданного выражения. В качестве значений аргумента A используем двоичное представление числа 15: 00001111, аргумента B — двоичную запись числа 51: 00110011, аргумента C — двоичную запись числа 85: 01010101.

Внимание! Была задана логическая функция от трех переменных. Следовательно, чтобы учесть все возможные сочетания значений переменных, при записи двоичного представления аргументов необходимо учитывать ведущие нули в двоичном представлении чисел 15, 51 и 85.

A	B	C	$A \leftrightarrow B$ (1)	$(1) \rightarrow C$ (2)	$\overline{(2)}$ (3)	$A \wedge C$ (4)	$\overline{(4)}$ (5)	$F = (3) \vee (5)$
0	0	0	1	0	1	0	1	1
0	0	1	1	1	0	0	1	1
0	1	0	0	1	0	0	1	1
0	1	1	0	1	0	0	1	1
1	0	0	0	1	0	0	1	1
1	0	1	0	1	0	1	0	0
1	1	0	1	0	1	0	1	1
1	1	1	1	1	0	1	0	0

Выписывая из этой таблицы значение функции F (сверху вниз от старшего разряда к младшему), получим двоичное число 11111010, которое соответствует числу 250 в десятичной системе счисления.

2-й способ.

Упростим исходное выражение:

$$\begin{aligned}
 & ((A \leftrightarrow B) \rightarrow C) \vee (A \wedge C) = \{\text{раскрываем эквиваленцию, применяем} \\
 & \text{закон де Моргана}\} = \overline{((\overline{A \vee B}) \wedge (A \vee \overline{B})) \vee C} \vee (\overline{A} \vee \overline{C}) = \\
 & = \overline{((\overline{A} \wedge \overline{B} \vee A \wedge B)) \rightarrow C} \vee (\overline{A} \vee \overline{C}) = \{\text{раскрываем импликацию}\} = \\
 & = \overline{((\overline{A} \wedge \overline{B} \vee A \wedge B)) \vee C \vee (\overline{A} \vee \overline{C})} = \{\text{закон де Моргана}\} = ((\overline{A} \wedge \overline{B} \vee A \wedge B)) \wedge \\
 & \wedge \overline{C} \vee (\overline{A} \vee \overline{C}) = \{\text{раскрываем скобки}\} = \overline{A} \wedge \overline{B} \wedge \overline{C} \vee A \wedge B \wedge \overline{C} \vee \overline{A} \vee \overline{C} = \\
 & = \overline{C} \wedge (A \wedge B \vee 1) \vee \overline{A} \wedge (\overline{B} \wedge \overline{C} \vee 1) \vee \overline{A} \vee \overline{C} = \overline{C} \wedge 1 \vee \overline{A} \wedge 1 \vee \overline{A} \vee \overline{C} = \overline{A} \vee \overline{C}.
 \end{aligned}$$

Таким образом, после упрощения получаем выражение:

$$((A \leftrightarrow B) \rightarrow C) \vee (A \wedge C) = \overline{A} \vee \overline{C}.$$

Запишем таблицу истинности для упрощенного выражения. В качестве значений аргумента A используем двоичное представление числа 15: 00001111, аргумента B — двоичную запись числа 51: 00110011, аргумента C — двоичную запись числа 85: 01010101.

A	B	C	\overline{A}	\overline{C}	$F = \overline{A} \vee \overline{C}$
0	0	0	1	1	1
0	0	1	1	0	1
0	1	0	1	1	1
0	1	1	1	0	1
1	0	0	0	1	1
1	0	1	0	0	0
1	1	0	0	1	1
1	1	1	0	0	0

Выписывая из таблицы значение функции F (сверху вниз от старшего разряда к младшему), получим двоичное число 11111010, которое соответствует числу 250 в десятичной системе счисления.

Ответ: 250.

2.4. Представление логических формул в нормальной форме

При решении задач часто требуется по таблице истинности логической формулы записать ее аналитическое выражение. Для этого используются понятия *совершенной дизъюнктивной нормальной формы (СДНФ)* и *совершенной конъюнктивной нормальной формы (СКНФ)*.

Простой конъюнкцией называется конъюнкция одной или нескольких переменных, в которой каждая переменная встречается не более одного раза (либо сама, либо ее отрицание). Например, запись $A \wedge B \wedge \bar{C}$ является простой конъюнкцией.

Простой дизъюнкцией называется дизъюнкция одной или нескольких переменных, в которую каждая переменная входит не более одного раза (либо сама, либо ее отрицание). Например, выражение $A \vee B \vee C$ — простая дизъюнкция.

Дизъюнктивной нормальной формой (ДНФ) называется дизъюнкция простых конъюнкций. Например, выражение $A \wedge B \vee C \wedge D$ является ДНФ.

Конъюнктивной нормальной формой (КНФ) называется конъюнкция простых дизъюнкций. Например, выражение $(A \vee B \vee \bar{C}) \wedge (A \vee C) \wedge (B \vee \bar{C})$ является КНФ.

Совершенной дизъюнктивной нормальной формой (СДНФ) называется такая дизъюнктивная нормальная форма, у которой в каждую конъюнкцию входят *все* переменные (либо сами, либо их отрицания). Например, выражение $A \vee \bar{B} \wedge \bar{C}$ является ДНФ, но не СДНФ. Выражение же $(A \wedge B \wedge C) \vee (A \wedge \bar{B} \wedge \bar{C}) \vee (A \wedge B \wedge C)$ представляет собой СДНФ.

Совершенной конъюнктивной нормальной формой (СКНФ) называется такая КНФ, у которой в каждую простую дизъюнкцию входят все переменные (либо сами, либо их отрицания). Например, выражение $(A \vee B \vee \bar{C}) \wedge (A \vee B \vee C) \wedge (A \vee \bar{B} \vee C)$ представляет собой СКНФ.

Пусть задана функция в ДНФ от трех переменных A, B, C :

$$F = (\bar{A} \wedge B) \vee (B \wedge \bar{C}).$$

Составим для заданной функции таблицу истинности:

№	A	B	C	\bar{A}	\bar{C}	$\bar{A} \wedge B$	$B \wedge \bar{C}$	F
1	0	0	0	1	1	0	0	0
2	0	0	1	1	0	0	0	0
3	0	1	0	1	1	1	1	1
4	0	1	1	1	0	1	0	1
5	1	0	0	0	1	0	0	0
6	1	0	1	0	0	0	0	0
7	1	1	0	0	1	0	1	1
8	1	1	1	0	0	0	0	0

Для представления заданной функции в СКНФ выберем из таблицы наборы значений переменных, при которых функция принимает значение 0. При записи СКНФ необходимо учитывать, что если значение логической переменной в рассматриваемом наборе равно 1, то она входит в дизъюнкцию с инверсией. Это наборы 1, 2, 5, 6 и 8. Например, для набора 5 значение переменной A равно 1, следовательно, она входит в дизъюнкцию с инверсией (\bar{A}), а значения переменных B и C равны 0, поэтому они входят в дизъюнкцию без инверсии. Следовательно, на основе набора 5 можно получить запись: $\bar{A} \vee B \vee C$.

В результате мы, соединяя такие наборы операцией «И», получим запись заданной функции в СКНФ: $(\bar{A} \vee B \vee C) \wedge (\bar{A} \vee B \vee C) \wedge (\bar{A} \vee B \vee C) \wedge (\bar{A} \vee B \vee C) \wedge (\bar{A} \vee B \vee C)$.

Для получения же СДНФ нужно выбрать из таблицы наборы переменных, при которых функция принимает значение 1, и соединить все такие наборы операцией «ИЛИ». При этом необходимо учитывать, что если значение логической переменной в рассматриваемом наборе равно 0, то она входит в конъюнкцию с инверсией. Это наборы 3, 4 и 7. Получим запись заданной функции в СДНФ: $(A \wedge B \wedge C) \vee (A \wedge B \wedge C) \vee (A \wedge B \wedge C)$.

Используя основные законы алгебры логики, нетрудно полученные выражения, записанные в СКНФ и СДНФ, привести к исходной форме. Выполним проверку. Упростим исходное выражение:

$$(A \wedge B) \vee (B \wedge C) = B \wedge (A \vee C).$$

Упростим СДНФ выражения:

$$\begin{aligned} (\bar{A} \wedge B \wedge \bar{C}) \vee (\bar{A} \wedge B \wedge C) \vee (A \wedge B \wedge \bar{C}) &= \{\text{вынесем из первых двух скобок } (\bar{A} \wedge B)\} = (\bar{A} \wedge B) \vee (C \wedge C) \vee (A \wedge B \wedge \bar{C}) = (A \wedge B) \vee (A \wedge B \wedge C) = \\ &= B \wedge (\bar{A} \vee A \wedge C) = \{\text{распределительный закон}\} = \\ &= B \wedge (\bar{A} \vee A \wedge \bar{C} \vee C) = B \wedge (\bar{A} \vee C). \end{aligned}$$

Упростим СКНФ выражения:

$$\begin{aligned}
& (A \vee B \vee C) \wedge (A \vee B \vee \bar{C}) \wedge (\bar{A} \vee B \vee C) \wedge (\bar{A} \vee B \vee \bar{C}) \wedge (\bar{A} \vee \bar{B} \vee \bar{C}) = \\
& = (A \vee B) \wedge (C \vee \bar{C}) \wedge (\bar{A} \vee B \vee C) \wedge (\bar{A} \vee \bar{C}) \wedge (B \vee \bar{B}) = \\
& = (A \vee B) \wedge (\bar{A} \vee B \vee C) \wedge (\bar{A} \vee \bar{C}) = (B \vee (A \wedge (\bar{A} \vee \bar{C}))) \wedge (\bar{A} \vee \bar{C}) = \\
& = (B \vee (A \wedge \bar{A} \vee \bar{A} \wedge C)) \wedge (\bar{A} \vee \bar{C}) = (B \vee \bar{A} \wedge C) \wedge (\bar{A} \vee \bar{C}) = \\
& = B \wedge \bar{A} \vee \bar{A} \wedge C \wedge A \vee B \wedge \bar{C} \vee \bar{A} \wedge C \wedge \bar{C} = B \wedge \bar{A} \vee B \wedge \bar{C} = B \wedge (\bar{A} \vee \bar{C}).
\end{aligned}$$

Задача. Дана таблица истинности выражения F :

X	Y	Z	F
1	0	1	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

Определите, соответствует ли F выражению

$$X \wedge \neg Y \wedge Z \vee \neg X \wedge Y \wedge Z \vee X \wedge Y \wedge \neg Z.$$

Решение.

1-й способ.

Подставим каждую строку таблицы истинности в приведенное в условии выражение и сравним полученный результат с значением логической функции.

- 1) $X = 1, Y = 0, Z = 1$: $(1 \wedge \bar{0} \wedge 1) \vee (\bar{1} \wedge 0 \wedge 1) \vee (1 \wedge 0 \wedge \bar{1}) = 1, F = 1$;
- 2) $X = 0, Y = 0, Z = 1$: $(0 \wedge \bar{0} \wedge 1) \vee (\bar{0} \wedge 0 \wedge 1) \vee (0 \wedge 0 \wedge \bar{1}) = 0, F = 0$;
- 3) $X = 0, Y = 1, Z = 0$: $(0 \wedge \bar{1} \wedge 0) \vee (\bar{0} \wedge 1 \wedge 0) \vee (0 \wedge 1 \wedge \bar{0}) = 0, F = 0$;
- 4) $X = 0, Y = 1, Z = 1$: $(0 \wedge \bar{1} \wedge 1) \vee (\bar{0} \wedge 1 \wedge 1) \vee (0 \wedge 1 \wedge \bar{1}) = 1, F = 1$;
- 5) $X = 1, Y = 0, Z = 0$: $(1 \wedge \bar{0} \wedge 0) \vee (\bar{1} \wedge 0 \wedge 0) \vee (1 \wedge 0 \wedge \bar{0}) = 0, F = 0$;
- 6) $X = 0, Y = 0, Z = 1$: $(1 \wedge \bar{0} \wedge 1) \vee (\bar{1} \wedge 0 \wedge 1) \vee (1 \wedge 0 \wedge \bar{1}) = 0, F = 0$;
- 7) $X = 1, Y = 1, Z = 0$: $(1 \wedge \bar{1} \wedge 0) \vee (\bar{1} \wedge 1 \wedge 0) \vee (1 \wedge 1 \wedge \bar{0}) = 1, F = 1$;
- 8) $X = 1, Y = 1, Z = 1$: $(1 \wedge \bar{1} \wedge 1) \vee (\bar{1} \wedge 1 \wedge 1) \vee (1 \wedge 1 \wedge \bar{1}) = 0, F = 0$.

Таким образом, таблица истинности указанного выражения соответствует таблице истинности выражения F .

2-й способ.

Таблица истинности построена для логической функции трех переменных. В этой таблице перечислены **все** сочетания значений

трех переменных (8 сочетаний), поэтому по данной таблице можно построить СКНФ или СДНФ функции, а затем сравнить полученную функцию с функцией, заданной в условии. (Если же в таблице истинности не указаны все возможные сочетания значений логических переменных, то по такой таблице построить СКНФ или СДНФ нельзя!)

СДНФ записывается по наборам переменных, для которых значение функции равно 1, а СКНФ — по наборам, для которых значение функции равно 0. Учитывая, что F принимает истинные значения на трех наборах переменных и в каждом варианте ответа также присутствуют три слагаемых, удобнее записывать СДНФ функции. (Конечно, можно записать и СКНФ, но тогда ее придется преобразовывать для получения трех слагаемых.)

При записи СДНФ выберем из таблицы истинности наборы переменных, для которых функция принимает значение 1. Если значение логической переменной в этом наборе равно 0, то в конъюнкцию запишем ее с инверсией. Получим следующую СДНФ:

$$F(X, Y, Z) = (X \wedge \bar{Y} \wedge Z) \vee (\bar{X} \wedge Y \wedge Z) \vee (X \wedge Y \wedge \bar{Z}).$$

Таким образом, данная функция соответствует приведенному в условии выражению.

Ответ: данная функция соответствует приведенному выражению.

2.5. Логические схемы

2.5.1. Переключательная схема

ЭВМ и другие устройства содержат в себе различные электрические схемы, содержащие переключательные элементы: реле, выключатели и т. п. Для разработки таких схем может быть использована *алгебра логики*.

Переключательная схема — это схематическое изображение устройства, состоящего из переключателей и соединяющих их проводников, а также из входов и выходов, на которые подается и с которых снимается электрический сигнал.

Каждый переключатель имеет только два состояния: замкнутое и разомкнутое. Переключателю A можно поставить в соответствие логическую переменную a . Эта переменная a принимает значение 1,

когда переключатель A замкнут (схема проводит ток), а если переключатель разомкнут, то a равна нулю.

Пусть два переключателя A и \bar{A} связаны так, что когда A замкнут, то \bar{A} разомкнут, и наоборот. Тогда если переключателю A поставлена в соответствие логическая переменная a , то переключателю \bar{A} должна соответствовать переменная \bar{a} .

Переключательной схеме можно поставить в соответствие функцию от переменных, соответствующих всем переключателям схемы. Такая функция называется *функцией проводимости*.

Запишем функции проводимости F для некоторых переключательных схем:

а) схема содержит один постоянно разомкнутый контакт: $F = 0$ (см. рис. 2.2, а);

б) схема не содержит переключателей и проводит ток всегда: $F = 1$ (см. рис. 2.2, б);

в) схема проводит ток, когда переключатель A замкнут, и не проводит, когда A разомкнут: $F(A) = A$ (см. рис. 2.2, в);

г) схема проводит ток, когда переключатель A разомкнут, и не проводит, когда A замкнут: $F(A) = \bar{A}$ (см. рис. 2.2, г);

д) схема проводит ток, когда оба переключателя замкнуты: $F(A) = A \wedge B$ (см. рис. 2.2, д);

е) схема проводит ток, когда хотя бы один из переключателей замкнут: $F(A) = A \vee B$ (см. рис. 2.2, е).

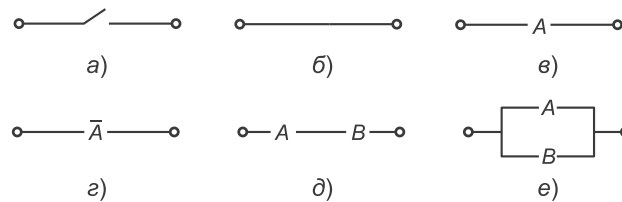
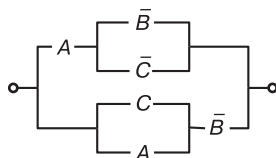


Рис. 2.2. Элементарные переключательные схемы

Две схемы называются *равносильными*, если через одну из них проходит ток тогда и только тогда, когда он проходит через другую при одном и том же входном сигнале. Из двух равносильных схем более простой считается схема, функция проводимости которой содержит меньшее число логических операций или переключателей.

Задача 1. Запишите по приведенной ниже переключательной схеме функцию проводимости.

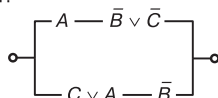


Решение.

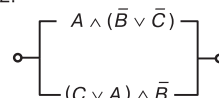
Указанная схема состоит из двух параллельных ветвей. Каждая из этих ветвей содержит последовательно соединенные элементы. Последовательное соединение соответствует логической операции конъюнкции, а параллельное — дизъюнкции.

Выполним преобразование схемы:

Шаг 1:



Шаг 2:



Учитывая приведенные схемы, запишем функцию проводимости в виде:

$$F(A, B, C) = A \wedge (\bar{B} \vee \bar{C}) \vee (C \wedge A) \wedge \bar{B} = A \wedge \bar{B} \vee A \wedge \bar{C} \vee C \wedge \bar{B} \vee A \wedge \bar{B} = A \wedge \bar{C} \vee \bar{B} \wedge (A \vee C).$$

$$\text{Ответ: } F(A, B, C) = A \wedge \bar{C} \vee \bar{B} \wedge (A \vee C).$$

2.5.2. Комбинационная схема

Логический элемент компьютера — часть электронной логической схемы, которая реализует элементарную логическую функцию.

Логическими элементами компьютеров являются *вентили* — электронные схемы «И», «ИЛИ», «НЕ», «И – НЕ», «ИЛИ – НЕ», а также *триггер*. Обычно у вентиля бывает от двух до восьми входов и один или два выхода. Представление логических состояний (1 и 0) в вентилях основано на использовании двух установленных уровней напряжения (например, +5 и 0). Высокий уровень напряжения обычно соответствует значению «истина» (1), а низкий — значению «ложь» (0).

С помощью логических элементов можно реализовать любую логическую функцию, описывающую работу устройств компьютера в виде *комбинационной схемы* — схемы, состоящей из логических

элементов, а также из входов и выходов, на которые подается и с которых считывается сигнал.

Каждый логический элемент имеет свое условное обозначение, которое выражает его логическую функцию (рис. 2.3).

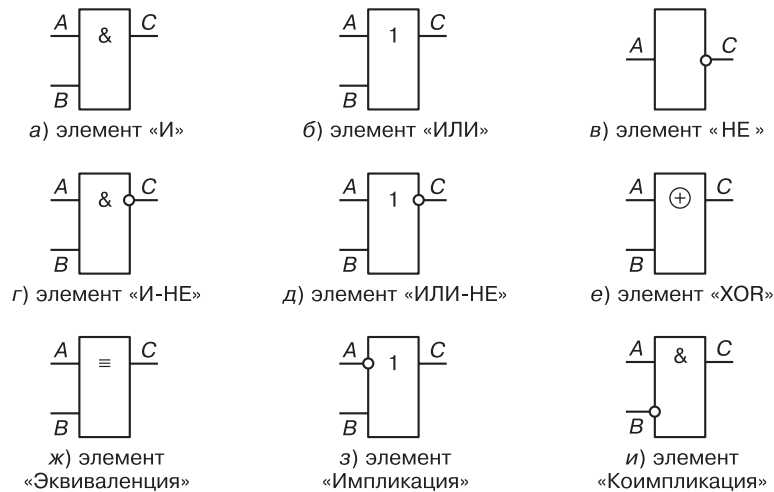


Рис. 2.3. Логические элементы

Элемент «И» реализует конъюнкцию двух или более логических значений. Условное обозначение на структурных схемах схемы «И» с двумя входами представлено на рис. 2.3, а. Операция конъюнкции на структурных схемах обозначается знаком «&». Связь между выходом этой схемы C и ее входами A и B описывается выражением $C = A \wedge B$.

Элемент «ИЛИ» реализует дизъюнкцию двух или более логических значений. Условное обозначение на структурных схемах схемы «ИЛИ» с двумя входами представлено на рис. 2.3, б. Связь между выходом C этой схемы и входами A и B описывается выражением $C = A \vee B$.

Элемент «НЕ» (инвертор, см. рис. 2.3, в) реализует операцию отрицания. Связь между входом A этой схемы и выходом C можно записать соотношением $C = \overline{A}$.

Элемент «И-НЕ» (элемент Шеффера, см. рис. 2.3, г) состоит из элемента «И» и инвертора. Данный элемент осуществляет отрицание результата действия элемента «И». Связь между выходом C и входами A и B этой схемы записывают выражением $C = \overline{A \wedge B}$.

Таблица истинности схемы «И–НЕ»

A	B	$\overline{A \wedge B}$
0	0	1
0	1	1
1	0	1
1	1	0

Элемент «ИЛИ–НЕ» (элемент Вебба, см. рис. 2.3, д) состоит из элемента «ИЛИ» и инвертора. Данный элемент осуществляет отрицание результата работы схемы «ИЛИ». Связь между выходом C и входами A и B этой схемы записывают выражением $C = A \vee B$.

Таблица истинности схемы «ИЛИ–НЕ»

A	B	$\overline{A \vee B}$
0	0	1
0	1	0
1	0	0
1	1	0

Элемент «XOR» (см. рис. 2.3, е) реализует операцию «сумма по модулю 2» (другое возможное обозначение этой функции — «М2»). Связь между выходом C и входами A и B этой схемы записывают выражением $C = A \oplus B$.

Элемент « \equiv » (см. рис. 2.3, ж) реализует операцию эквиваленции. Связь между выходом C и входами A и B этой схемы записывают выражением $C = A \equiv B$.

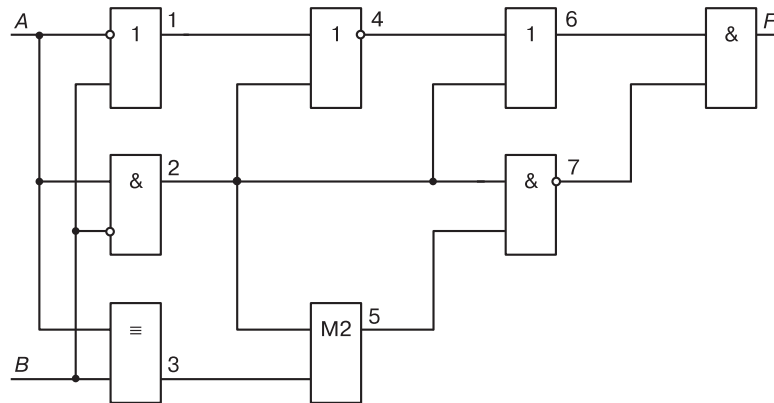
Элемент «Импликация» (см. рис. 2.3, з) реализует импликацию. Связь между выходом C и входами A и B этой схемы записывают выражением $C = A \rightarrow B$.

Элемент «Коимпликация» (см. рис. 2.3, и) реализует отрицание импликации. Связь между выходом C и входами A и B этой схемы записывают выражением $C = A \rightarrow B = A \wedge \overline{B}$.

Таблица истинности схемы «Коимпликация»

A	B	$A \wedge \overline{B}$
0	0	0
0	1	0
1	0	1
1	1	0

Задача 2. Определите логическую функцию, которую реализует комбинационная схема устройства.



Решение.

1-й способ.

Обозначим выходы блоков цифрами и запишем таблицы истинности выражений, реализуемые блоками (номера блоков расставлялись сверху вниз и слева направо):

		1	2	3	4	5	6	7	F
A	B	$\overline{A \vee B}$	$A \wedge \overline{B}$	$A \equiv B$	$\overline{1 \vee 2}$	$2 \oplus 3$	$4 \vee 2$	$\overline{2 \wedge 5}$	$6 \wedge 7$
0	0	1	0	1	0	1	0	1	0
0	1	1	0	0	0	0	0	1	0
1	0	0	1	0	0	1	1	0	0
1	1	1	0	1	0	1	0	1	0

Таким образом, данная схема реализует тождественно ложную логическую функцию.

2-й способ.

На вход первого элемента поступают величины A и B . В соответствии с операцией, реализуемой данным элементом, на его выходе появляется величина $\overline{A \vee B}$. Запишем выражения, соответствующие результатам работы каждого элемента (цифра обозначает номер элемента):

1) $\overline{A \vee B}$;

2) $A \wedge \overline{B}$;

3) $A \equiv B = \overline{A \wedge \overline{B}} \vee A \wedge B$;

4) $\overline{(\overline{A \vee B}) \vee (A \wedge \overline{B})} = \overline{\overline{A \vee B} \wedge A \wedge \overline{B}} = A \wedge \overline{B} \wedge (\overline{A \vee \overline{B}}) = A \wedge \overline{B} \wedge (\overline{A \vee B}) = A \wedge \overline{B} \wedge A \vee A \wedge \overline{B} \wedge B = 0 \vee 0 = 0$;

5) $(A \wedge B) \oplus (A \equiv B) = (A \wedge \overline{B}) \wedge (A \equiv B) \vee (A \wedge \overline{B}) \wedge \overline{(A \equiv B)}$.

Упростим отдельные части выражения:

$$\begin{aligned} \overline{(A \equiv B)} &= \overline{(\overline{A} \wedge \overline{B} \vee A \wedge B)} = \overline{(\overline{A} \wedge \overline{B} \wedge A \wedge B)} = (\overline{\overline{A}} \vee \overline{\overline{B}}) \wedge (\overline{A} \vee \overline{B}) = \\ &= (A \vee B) \wedge (\overline{A} \vee \overline{B}) = A \wedge \overline{A} \vee A \wedge \overline{B} \vee B \wedge \overline{A} \vee B \wedge \overline{B} = A \wedge \overline{B} \vee B \wedge \overline{A}; \\ \overline{(A \wedge B)} &= \overline{A} \vee \overline{B} = \overline{A} \vee B. \end{aligned}$$

Получим:

$$\begin{aligned} \overline{(A \wedge \overline{B})} \wedge (A \equiv B) \vee (A \wedge \overline{B}) \wedge \overline{(A \equiv B)} &= (\overline{A} \vee B) \wedge (\overline{A} \wedge \overline{B} \vee A \wedge B) \vee \\ &\vee (A \wedge \overline{B}) \wedge (A \wedge \overline{B} \vee B \wedge A) = A \wedge \overline{A} \wedge \overline{B} \vee B \wedge \overline{A} \wedge \overline{B} \vee \overline{A} \wedge A \wedge B \vee B \wedge \\ &\wedge A \wedge B \vee A \wedge \overline{B} \wedge A \wedge \overline{B} \vee A \wedge \overline{B} \wedge B \wedge A = \overline{A} \wedge \overline{B} \vee 0 \vee 0 \vee A \wedge B \vee A \wedge \\ &\wedge \overline{B} \vee 0 = \overline{A} \wedge \overline{B} \vee A \wedge B \vee A \wedge \overline{B} = \overline{A} \wedge \overline{B} \vee A \wedge (B \vee \overline{B}) = \overline{A} \wedge \overline{B} \vee A. \end{aligned}$$

$$6) 0 \vee A \wedge \overline{B} = A \wedge \overline{B};$$

$$\begin{aligned} 7) \overline{(A \wedge \overline{B})} \wedge (\overline{A} \wedge \overline{B} \vee A) &= \overline{(A \wedge \overline{B})} \vee \overline{(\overline{A} \wedge \overline{B} \vee A)} = (\overline{A} \vee B) \vee \\ &\vee (\overline{A} \wedge \overline{B} \wedge A) = (\overline{A} \vee B) \vee ((\overline{A} \vee \overline{B}) \wedge A) = (\overline{A} \vee B) \vee ((A \vee B) \wedge \overline{A}) = \\ &= \overline{A} \vee B \vee (A \wedge \overline{A} \vee B \wedge \overline{A}) = \overline{A} \vee B \vee B \wedge \overline{A} = \overline{A} \wedge (B \vee 1) \vee B = \overline{A} \vee B; \end{aligned}$$

$$8) F = (A \wedge \overline{B}) \wedge (\overline{A} \vee B) = A \wedge \overline{B} \wedge \overline{A} \vee A \wedge \overline{B} \wedge B = 0 \vee 0 = 0.$$

Таким образом, данная схема реализует тождественно ложную логическую функцию.

Ответ: $F = 0$.

2.6. Теория игр

Теория игр — это математический метод изучения оптимальных стратегий в играх. Впервые математические аспекты и приложения теории игр были изложены в книге Джона фон Неймана и Оскара Моргенштерна «Теория игр и экономического поведения» (1944 г.).

Под *игрой* в данном случае понимается некоторый процесс, в котором участвуют два и более игроков, ведущих борьбу за реализацию своих интересов. У каждого игрока при этом существует цель и стратегия, которая может привести к выигрышу или проигрышу в зависимости от поведения других игроков. Теория игр помогает выбрать лучшую стратегию с учетом представлений о других участниках, их ресурсах и возможных поступках.

Игрок побеждает в игре, если он имеет возможность ходить и отвечать на ходы противника таким образом, чтобы выиграть независимо от ходов противника. Эта возможность и описание того, как именно нужно отвечать на ходы соперника, называется *выигрышной стратегией*. Безошибочная игра заключается в том, что игрок не должен делать ходов, которые могут привести к выигрышу друго-

го игрока, при наличии других ходов, не приводящих к победе второго игрока.

Существует несколько форм представления игр. Одной из них является *экстенсивная*, или *расширенная, форма*. Игры в подобной форме представляются в виде дерева, где каждая вершина соответствует выбору тем или иным игроком своей стратегии. Возможным ходам игрока в этом дереве при этом обычно соответствует целый уровень вершин (см. рис. 2.4).

Задача 1. Два игрока играют в следующую игру. На клетчатой доске стоит фигура в точке с координатами $(2, 2)$.

Ход состоит в том, что игрок перемещает фигуру из точки с координатами (x, y) в одну из двух точек: или в точку с координатами $(x+1, y+2)$, или в точку с координатами $(x+2, y+1)$. Выигрывает игрок, после хода которого фигура окажется в точке $(8, 8)$. Если фигура окажется в точке $(8, y)$ или в точке $(x, 8)$, то считается, что победителя нет.

8								
7								
6								
5								
4								
3								
2		•						
1								
	1	2	3	4	5	6	7	8

Кто выигрывает при безошибочной игре обоих игроков — игрок, делающий первый ход, или игрок, делающий второй ход? Каким должен быть первый ход выигрывающего игрока? Ответ обоснуйте.

Решение.

Игрок может переместить фигуру из точки (x, y) в точку с координатами $(x+1, y+2)$ или $(x+2, y+1)$. Таким образом, у каждого игрока существует два возможных хода.

Сформируем дерево игры (рис. 2.4). В вершине дерева поместим запись « $(2, 2)$ », которая означает, что фигура находится в точке с координатами $(2, 2)$. Далее дерево строится до тех пор, пока в каждой

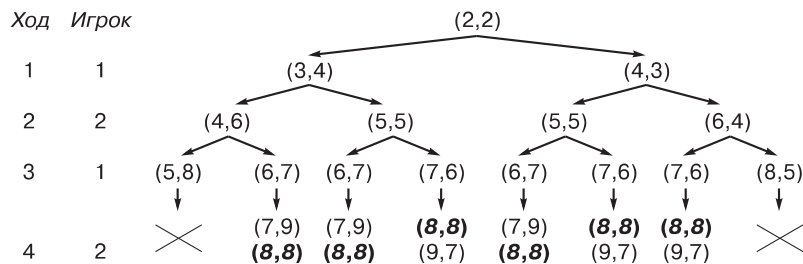


Рис. 2.4. Дерево игры

его ветке не будет получена хотя бы одна выигрышная комбинация или комбинация, в которой ни один из игроков не выигрывает (комбинация $(8, y)$ или $(x, 8)$).

Рассмотрим процесс игры. Допустим, что ход первого игрока — $(3, 4)$ (левая ветвь дерева на рис. 2.4), тогда у второго игрока возможны следующие варианты хода:

1) $(4, 6)$ — невыигрышная комбинация для второго игрока — если первый игрок на третьем ходе сводит игру к комбинации $(5, 8)$, то при ее получении считается, что победителя в игре нет. Тем самым первый игрок не победит, но и одновременно не даст победить второму игроку;

2) $(5, 5)$ — выигрышная комбинация для второго игрока — первый игрок на третьем ходе получает комбинацию $(6, 7)$ или $(7, 6)$, и **любую** из этих комбинаций второй игрок на четвертом ходе игры может свести к выигрышной комбинации $(8, 8)$.

Следовательно, при ходе первого игрока $(3, 4)$ второй игрок выигрывает на четвертом ходе, для чего на втором ходе игры он должен получить комбинацию $(5, 5)$.

Безошибочная игра второго игрока в данном случае заключается в получении комбинации $(5, 5)$, так как в противном случае первый игрок сводит игру к невыигрышной комбинации $(5, 8)$. Поэтому если второй игрок сделает ход $(4, 6)$, то его игра не будет считаться безошибочной.

Предположим теперь, что первый игрок переместил фишку в точку $(4, 3)$ (правая ветвь дерева на рис. 2.4), тогда у второго игрока возможны следующие варианты хода:

1) $(5, 5)$ — выигрышная комбинация для второго игрока (см. предыдущие рассуждения);

2) $(6, 4)$ — невыигрышная комбинация для второго игрока — если первый игрок на третьем ходе сводит игру к комбинации $(8, 5)$, то при ее получении считается, что победителя в игре нет.

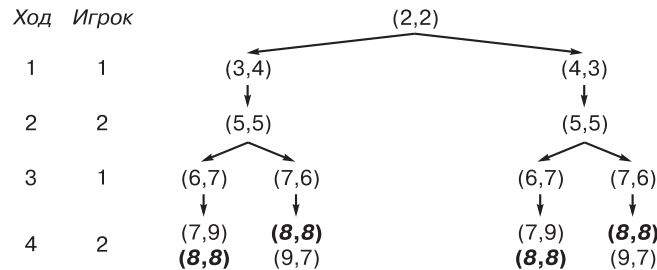
Следовательно, при ходе первого игрока $(6, 4)$ второй игрок выигрывает на четвертом ходе, для чего на втором ходе игры он должен получить комбинацию $(5, 5)$.

Таким образом, в данной игре победитель — второй игрок, поскольку показано, что он побеждает при **любом** первом ходе первого игрока.

Ответ:

1) выигрывает игрок, который делает второй ход;

2) следующее дерево игры доказывает, что выигрывает второй игрок;



3) первый ход выигрывающего игрока: второй игрок должен своим первым ходом получить комбинацию (5,5).

Задача 2. Два игрока играют в следующую игру. Перед ними лежат две кучки камней, в первой из которых 1, а во второй — 2 камня. У каждого игрока неограниченно много камней. Игроки ходят по очереди. Ход состоит в том, что игрок или увеличивает в 3 раза количество камней в какой-то куче, или добавляет 2 камня в какую-то кучу. Выигрывает игрок, после хода которого общее число камней в двух кучах становится не менее 17. Кто выигрывает при безошибочной игре обоих игроков — игрок, делающий первый ход, или игрок, делающий второй ход? Каким должен быть первый ход выигрывающего игрока? Ответ обоснуйте [5].

Решение.

Сформируем дерево игры (см. рис. 2.5). В вершине дерева поместим запись «(1, 2)», которая означает, что в первой кучке находится 1 камень, а во второй — 2 камня. Дерево строится до тех пор, пока в каждой ветке не будет получена хотя бы одна выигрышная комбинация. При этом игрок может увеличить в какой-то куче количество камней в 3 раза или добавить к какой-то куче 2 камня, а так как всего имеется две кучи камней, то у каждого игрока существует четыре возможных хода.

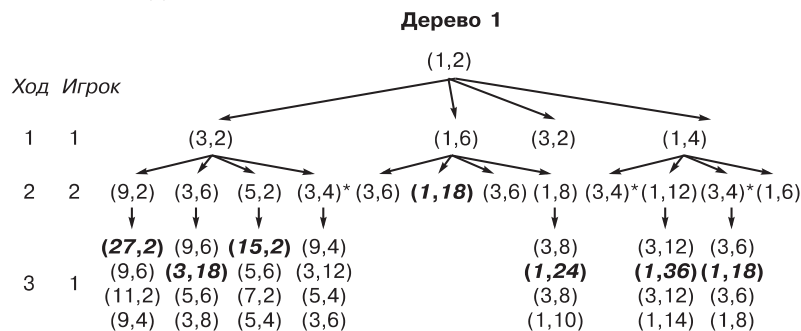


Рис. 2.5. Дерево игры

Проанализируем построенное дерево. На рис. 2.5 показана основная часть дерева игры, а на рис. 2.6 — его отдельная ветвь, исходящая из узла «(3,4)*» — дерево 2. Как нетрудно увидеть, некоторые пары, выделенные жирным шрифтом, в дереве повторяются несколько раз:

- 1) (3,2) → (9,2), (3,6), (5,2), (3,4);
- 2) (1,6) → (3,6), (1,18), (3,6), (1,8);
- 3) (3,2) → (9,2), (3,6), (5,2), (3,4);
- 4) (1,4) → (3,4), (1,12), (3,4), (1,6),

поэтому последующие их разложения в дерево решения можно не записывать.



Рис. 2.6. Дерево игры 2

I. Допустим, ход первого игрока — (3,2). Тогда у второго игрока возможны следующие варианты хода:

1) (9,2) — невыигрышная комбинация для второго игрока — на третьем ходе игры первый игрок получает выигрышную комбинацию (27,2);

2) (3,6) — невыигрышная комбинация для второго игрока — на третьем ходе игры первый игрок получает выигрышную комбинацию (3,18);

3) (5,2) — невыигрышная комбинация для второго игрока — на третьем ходе игры первый игрок сводит игру к выигрышной комбинации (15,2);

4) (3,4) — выигрышная комбинация для второго игрока — на третьем ходе у первого игрока нет ни одной выигрышной комбинации, а на четвертом ходе игры второй игрок сводит любую комбинацию первого игрока к выигрышной (см. дерево 2 на рис. 2.6).

Таким образом, при ходе первого игрока (3,2) второй игрок побеждает в игре на четвертом ходе, для чего своим первым ходом он должен получить комбинацию (3,4).

II. Предположим, что ход первого игрока — $(1,6)$, тогда второй игрок может сделать один из следующих ходов:

- 1) $(3,6)$ — невыигрышная комбинация для второго игрока;
- 2) $(1,18)$ — выигрышная комбинация для второго игрока;
- 3) $(3,6)$ — невыигрышная комбинация для второго игрока;
- 4) $(1,8)$ — невыигрышная комбинация для второго игрока — первый игрок на третьем ходе получает выигрышную комбинацию $(1,24)$.

Вывод: при ходе первого игрока $(1,6)$ второй игрок побеждает в игре на втором ходе, получая комбинацию $(1,18)$.

III. Предположим, что ход первого игрока — $(1,4)$, тогда второй игрок может сделать следующие ходы:

- 1) $(3,4)$ — выигрышная комбинация для второго игрока;
- 2) $(1,12)$ — невыигрышная комбинация для второго игрока — на третьем ходе игры первый игрок получает выигрышную комбинацию $(1,36)$;
- 3) $(1,6)$ — невыигрышная комбинация для второго игрока — на третьем ходе игры первый игрок получает выигрышную комбинацию $(1,18)$.

Следовательно, при ходе первого игрока $(1,4)$ второй игрок побеждает в игре на четвертом ходе, для чего своим первым ходом он должен получить комбинацию $(3,4)$.

Таким образом, в начале игры у первого игрока есть четыре возможных хода: $(3,2)$, $(1,6)$, $(3,2)$ и $(1,4)$, каждый из которых второй игрок может свести к комбинации $(3,4)$ в первом, третьем или четвертом случаях или к комбинации $(1,18)$ во втором случае, что позволит ему выиграть на четвертом или на втором ходе игры.

Ответ:

- 1) выигрывает игрок, который делает второй ход;
- 2) следующее дерево игры доказывает, что выигрывает второй игрок:

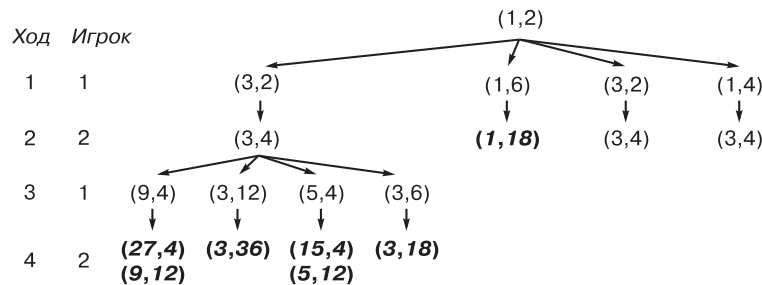


Рис. 2.7. Дерево игры

3) первый ход выигрывающего игрока: второй игрок должен своим первым ходом получить комбинацию (3,4) или (1,18).

Задача 3. Два игрока играют в следующую игру. На координатной плоскости стоит фишка. В начале игры фишка находится в точке с координатами $(-2, -1)$. Игроки ходят по очереди. Ход состоит в том, что игрок перемещает фишку из точки с координатами (x, y) в одну из трех точек: $(x + 3, y)$, $(x, y + 4)$ или $(x + 2, y + 2)$. Игра заканчивается, как только расстояние от фишки до начала координат превысит число 9. Выигрывает игрок, который сделал последний ход. Кто выигрывает при безошибочной игре — игрок, делающий первый ход, или игрок, делающий второй ход? Каким должен быть первый ход выигрывающего игрока? Ответ обоснуйте [7].

Решение.

Расстояние d между точкой A с координатами (x_1, y_1) и точкой B с координатами (x_2, y_2) вычисляется по формуле:

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}.$$

Если точка B представляет собой начало координат, то эту формулу можно записать в виде:

$$d = \sqrt{x_1^2 + y_1^2}.$$

Чтобы упростить вычисления, можно вычислять не расстояние d от фишки до начала координат, а квадрат данного расстояния:

$$d^2 = x_1^2 + y_1^2.$$

Сформируем дерево игры (см. рис. 2.8). При его записи вместо расстояния от фишки до начала координат указывается квадрат расстояния. Игра заканчивается, если квадрат расстояния от фишки до начала координат превысит число 81.

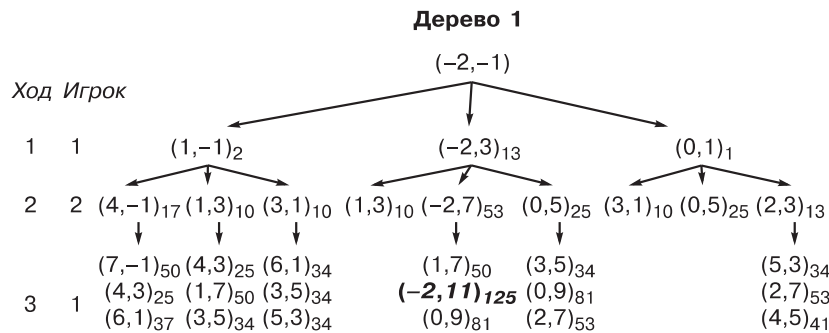


Рис. 2.8. Дерево 1

Игрок может переместить фишку в одну из трех точек: $(x + 3, y)$, $(x, y + 4)$ или $(x + 2, y + 2)$. Следовательно, у каждого игрока существует три возможных хода.

Проанализируем построенное дерево.

Запись « $(1, 7)_{50}$ » при этом означает, что фишка находится в точке с координатами $(1, 7)$, а квадрат расстояния от начала координат до данной точки равен 50.

I. Допустим, ход первого игрока — $(1, -1)$, тогда возможны три варианта хода второго игрока*:

1) $(4, -1)$ — невыигрышная комбинация для второго игрока — на третьем ходе игры первый игрок сводит игру к комбинации $(4, 3)$, в результате чего на четвертом ходе у второго игрока нет ни одной выигрышной комбинации (см. рис. 2.9);

Ход Игрок

Дерево 2.1

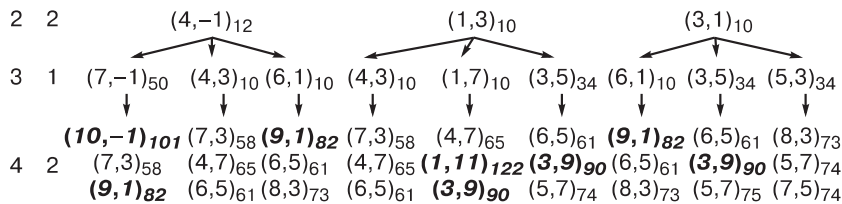


Рис. 2.9. Дерево 2.1

2) $(1, 3)$ — невыигрышная комбинация для второго игрока (рассуждения те же, что для комбинации $(4, -1)$ — см. рис. 2.9);

3) $(3, 1)$ — невыигрышная комбинация для второго игрока — на третьем ходе игры первый игрок сводит игру к комбинации $(5, 3)$, в результате чего на четвертом ходе у второго игрока нет ни одного выигрышного хода (см. рис. 2.9).

Ход Игрок

Дерево 2.2

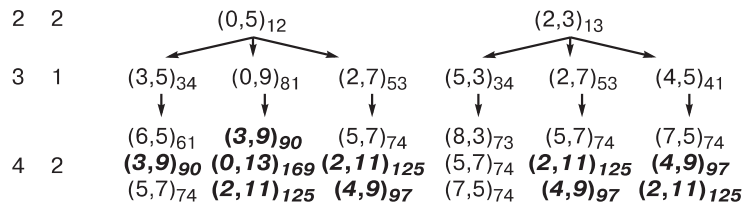


Рис. 2.10. Дерево 2.2

* Необходимо отметить, что комбинации $(3, 1)$ и $(1, 3)$ не являются эквивалентными.

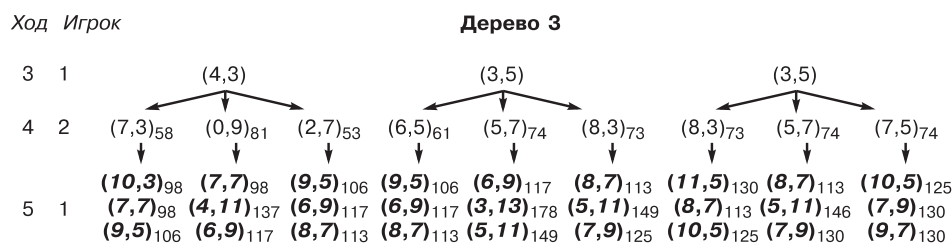


Рис. 2.11. Дерево 3

Таким образом, если ход первого игрока — $(1, -1)$, то он *всегда* будет выигрывать в дальнейшем.

II. Допустим, первый игрок своим первым ходом получает комбинацию $(-2, 3)$. В результате у второго игрока возможны следующие варианты хода:

1) $(1, 3)$ — невыигрышная комбинация для второго игрока — на третьем ходе первый игрок сводит игру к комбинации $(4, 3)$, из которой на четвертом ходе у второго игрока нет выигрышных ходов, но выигрышный ход есть на 5-м ходе у первого игрока;

2) $(-2, 7)$ — невыигрышная комбинация для второго игрока — первый игрок на третьем ходе игры получает выигрышную комбинацию $(-2, 11)$;

3) $(0, 5)$ — выигрышная комбинация для второго игрока — вне зависимости от хода первого игрока на 3-м шаге второй игрок может свести его к выигрышной комбинации.

Следовательно, первый игрок не должен делать ход $(-2, 3)$, так как это противоречит выигрышной стратегии для данного игрока, иначе второй игрок, получив комбинацию $(0, 5)$, выиграет на 4-м ходе.

III. Допустим, ход первого игрока — $(0, 1)$. Рассмотрим возможные варианты хода второго игрока:

1) $(3, 1)$ — невыигрышная комбинация для второго игрока — на третьем ходе игры первый игрок может свести игру к комбинации $(5, 3)$, в результате чего на четвертом ходе у второго игрока нет ни одной выигрышной комбинации (см. рис. 2.9);

2) $(0, 5)$ — второй игрок, получив такую комбинацию, выигрывает на 4-м ходе;

3) $(2, 3)$ — невыигрышная комбинация для второго игрока — на третьем ходе первый игрок получает комбинацию $(5, 3)$ (см. рис. 2.10).

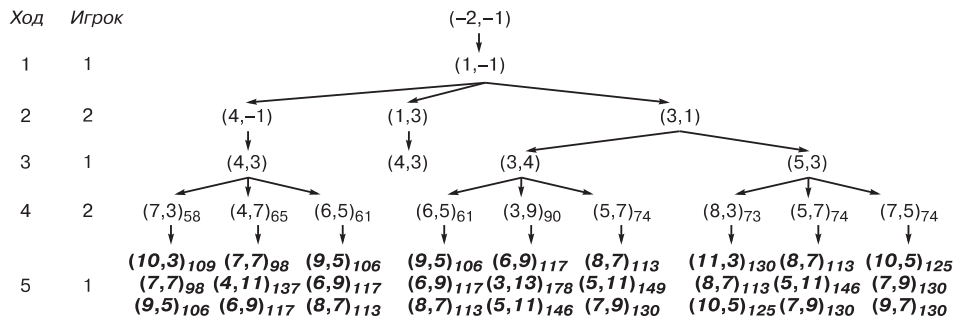
В данном случае (при ходе первого игрока $(0, 1)$) в двух вариантах из трех выигрывает первый игрок. Однако так как второй ход —

у второго игрока, то он, придерживаясь своей стратегии, может получить комбинацию $(0,5)$ и выиграть на четвертом ходе. Учитывая же, что первый игрок ходит безошибочно и преследует цель победить в игре, он должен своим первым ходом получить комбинацию $(1,-1)$, и тогда ему гарантирован выигрыш.

Ответ:

1) выигрывает игрок, который делает первый ход;

2) следующее дерево игры доказывает, что выигрывает первый игрок:



3) первый ход выигрывающего игрока: первый игрок должен своим первым ходом получить комбинацию $(1,-1)$.

Задача 4. Два игрока играют в следующую игру. Перед ними лежат три кучки камней, содержащих соответственно 2, 3 и 4 камня. У каждого игрока неограниченно много камней. Игроки ходят по очереди. Ход состоит в том, что игрок или увеличивает в 2 раза количество камней в меньшей куче (если их две — то в каждой из них), или добавляет по 2 камня в каждую из трех куч. Выигрывает игрок, после хода которого общее число камней в трех кучах становится не менее 26. Кто выигрывает при безошибочной игре обоих игроков — игрок, делающий первый ход, или игрок, делающий второй ход? Каким должен быть первый ход выигрывающего игрока? Ответ обоснуйте.

Решение.

Игрок может увеличить количество камней в меньшей куче в 2 раза или добавить в каждую кучу по 2 камня. Следовательно, у игрока каждый раз существует два возможных хода.

Сформируем дерево игры (см. рис. 2.12). В вершине дерева поместим запись « $(2,3,4)$ », которая означает, что в первой куче находится 2 камня, во второй — 3, а в третьей — 4 камня.

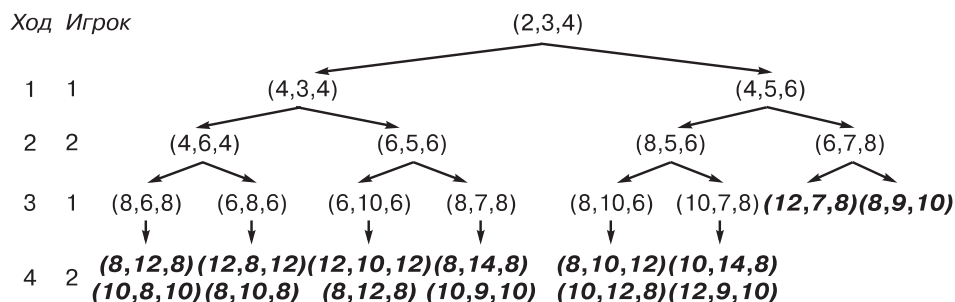


Рис. 2.12. Дерево игры

Проанализируем построенное дерево.

I. Допустим, ход первого игрока — $(4,3,4)$, тогда возможны два варианта хода второго игрока:

1) $(4,6,4)$ — выигрышный ход второго игрока — на третьем ходе первый игрок не может выиграть, а на четвертом ходе второй игрок независимо от хода первого игрока получает выигрышные комбинации $(8,12,8)$, $(10,8,10)$, $(12,8,12)$, $(8,10,8)$;

2) $(6,5,6)$ — выигрышный ход второго игрока — на третьем ходе первый игрок не может выиграть, а на четвертом ходе второй игрок независимо от хода первого игрока получает выигрышные комбинации $(12,10,12)$, $(8,12,8)$, $(10,9,10)$.

Таким образом, при ходе первого игрока $(4,3,4)$ второй игрок выигрывает на четвертом ходе игры, для чего на втором ходе игры он должен получить комбинацию $(4,6,4)$ или $(6,5,6)$.

II. Допустим, ход первого игрока — $(4,5,6)$, тогда возможны два варианта хода второго игрока:

1) $(8,5,6)$ — выигрышный ход второго игрока — на третьем ходе первый игрок не может выиграть, а на четвертом ходе второй игрок независимо от хода первого игрока получает выигрышные комбинации $(8,10,12)$, $(10,12,8)$, $(10,14,8)$, $(12,9,10)$;

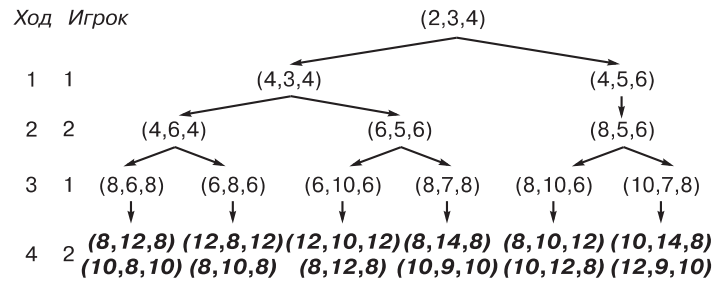
2) $(6,7,8)$ — невыигрышный ход второго игрока — независимо от хода второго игрока первый игрок на третьем ходе игры получает выигрышные комбинации $(12,7,8)$ и $(8,9,10)$.

Следовательно, при ходе первого игрока $(4,5,6)$ второй игрок выигрывает на четвертом ходе игры, для чего на втором ходе он должен получить комбинацию $(8,5,6)$.

Таким образом, второй игрок при реализации своей стратегии выигрывает в игре, получая своим первым ходом в игре одну из комбинаций: $(4,6,4)$, $(6,5,6)$ или $(8,5,6)$.

Ответ:

- 1) выигрывает игрок, который делает второй ход;
- 2) следующее дерево игры доказывает, что выигрывает второй игрок:



- 3) первый ход выигрывающего игрока: второй игрок должен своим первым ходом получить комбинации (4,6,4), (6,5,6) или (8,5,6).

Задача 5. Два игрока играют в следующую игру. На координатной плоскости стоит фишка. Игроки ходят по очереди. В начале игры фишка находится в точке с координатами (5, 2). Ход состоит в том, что игрок перемещает фишку из точки с координатами (x, y) в одну из трех точек: в точку с координатами $(x + 3, y)$, в точку с координатами $(x, y + 3)$ или в точку с координатами $(x, y + 4)$. Выигрывает игрок, после хода которого расстояние от фишки до точки с координатами (0, 0) не меньше 13 единиц. Кто выигрывает при безошибочной игре обоих игроков — игрок, делающий первый ход, или игрок, делающий второй ход? Каким должен быть первый ход выигрывающего игрока? Ответ обоснуйте.

Решение.

Сформируем дерево игры (см. рис. 2.13), используя тот же подход к вычислению расстояний, что и в задаче 3: вместо расстояния от фишки до начала координат будем вычислять квадрат этого расстояния.

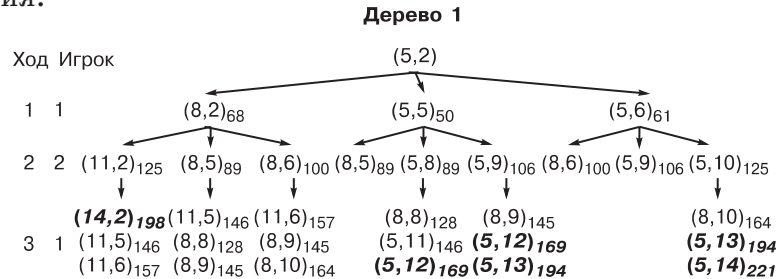


Рис. 2.13. Дерево игры

В вершине дерева поместим запись «(5,2)», которая означает, что фишка находится в точке с координатами (5,2). При записи дерева игры вместо расстояния от фишки до начала координат будет вычисляться его квадрат, следовательно, игра закончится, если квадрат расстояния от фишки до начала координат будет больше или равен 169.

Игрок может переместить фишку из точки координатами (x, y) в одну из трех точек: $(x + 3, y)$, $(x, y + 3)$ или $(x, y + 4)$. Следовательно, у игрока существует три возможных хода.

Проанализируем построенное дерево.

I. Допустим, ход первого игрока — (8,2), тогда возможны три варианта хода второго игрока:

1) (11,2) — невыигрышный ход второго игрока — на третьем ходе игры первый игрок получает выигрышную комбинацию (14,2);

2) (8,5) или (8,6) — выигрышный ход второго игрока — на третьем ходе игры первый игрок не может получить ни одной выигрышной комбинации, а на четвертом ходе игры второй игрок получает выигрышную комбинацию при любом ходе первого игрока (см. дерево 2 на рис. 2.14).

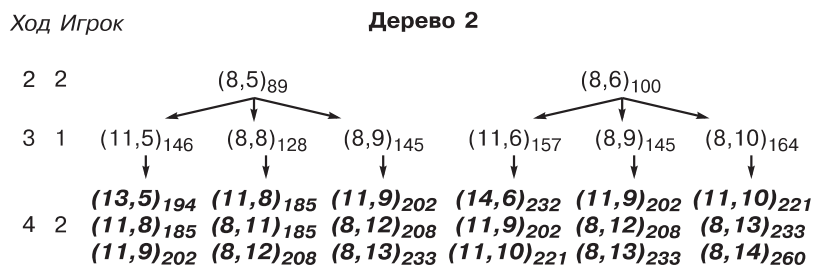


Рис. 2.14. Дерево 2

Таким образом, при ходе первого игрока (8,2) второй игрок выигрывает на четвертом ходе, для чего на втором ходе он должен получить комбинацию (8,5) или (8,6).

II. Предположим, что ход первого игрока — (5,5), тогда возможны следующие варианты хода второго игрока:

1) (8,5) — выигрышный ход второго игрока;

2) (5,8) — невыигрышный ход второго игрока — на третьем ходе первый игрок получает комбинацию (5,12);

3) (5,9) — невыигрышный ход второго игрока — на третьем ходе первый игрок получает комбинацию (5,13) или (5,14).

Следовательно, в данном случае в игре выигрывает второй игрок, делая ход (8,5).

III. Предположим, что ход первого игрока — (5,6), тогда возможны следующие варианты хода второго игрока:

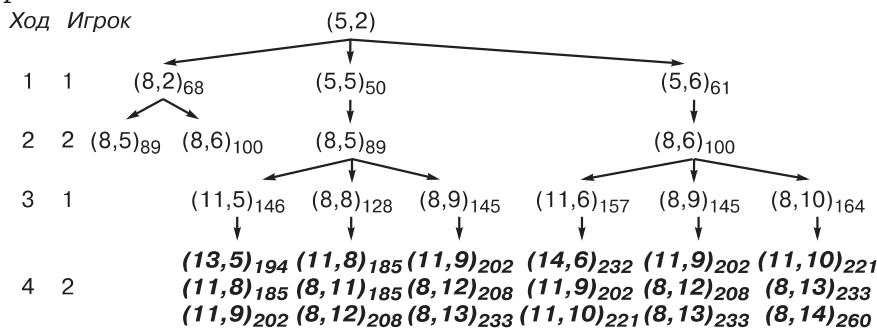
- 1) (8,6) — выигрышный ход второго игрока;
- 2) (5,9) — невыигрышный ход второго игрока;
- 3) (5,10) — невыигрышный ход второго игрока — на третьем ходе первый игрок получает выигрышную комбинацию (5,12) или (5,13).

Следовательно, в данном случае побеждает второй игрок, делая ход (8,6).

Таким образом, в игре побеждает второй игрок, так как именно он определяет второй ход и может вне зависимости от хода первого игрока свести всю игровую ситуацию к выгодным для него комбинациям (8,5) или (8,6) и в результате победить в игре.

Ответ:

- 1) выигрывает игрок, который делает второй ход;
- 2) следующее дерево игры доказывает, что выигрывает второй игрок:



- 3) первый ход выигрывающего игрока: второй игрок должен своим первым ходом получить комбинации (8,5) или (8,6).

Задача 6. Два игрока играют в следующую игру. Перед ними лежат три кучки камней, в первой из которых 2, во второй — 3, в третьей — 4 камня. У каждого игрока неограниченно много камней. Игроки ходят по очереди. Ход состоит в том, что игрок или удваивает количество камней в какой-то куче, или добавляет по два камня в каждую из куч. Выигрывает игрок, после хода которого либо в одной из куч становится не менее 15 камней, либо общее число камней во всех трех кучах становится не менее 25. Кто выигрывает при безошибочной игре обоих игроков — игрок, делающий первый ход, или игрок, делающий второй ход? Каким должен быть первый ход выигрывающего игрока? Ответ обоснуйте [1].

Решение.

Игрок может удвоить количество камней в 2 раза в какой-либо куче или добавить в каждую кучу по 2 камня. Следовательно, у игрока существует четыре возможных хода.

Сформируем дерево игры (см. рис. 2.15). В вершине дерева поместим запись «(2,3,4)», которая означает, что в первой куче находится 2 камня, во второй — 3, а в третьей — 4.

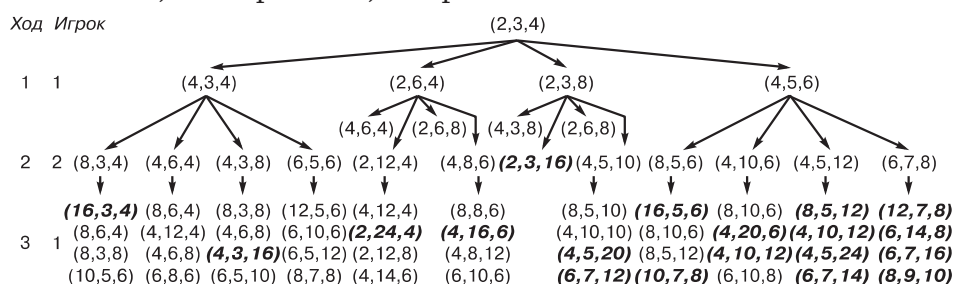


Рис. 2.15. Дерево игры

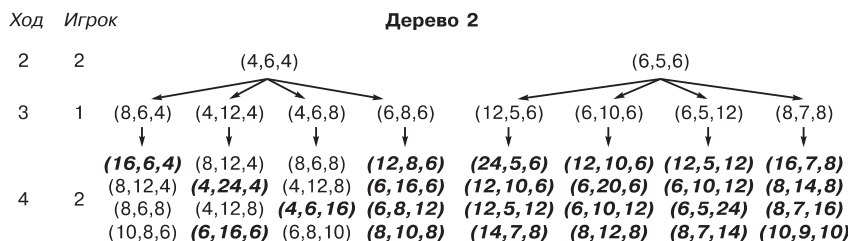


Рис. 2.16. Дерево 2

Проанализируем построенное дерево.

I. Допустим, ход первого игрока — (4,3,4), тогда возможны следующие варианты хода второго игрока:

1) (8,3,4) — невыигрышный ход для второго игрока — первый игрок выигрывает на третьем ходе, получив комбинацию (16,3,4);

2) (4,6,4) и (6,5,6) — выигрышные комбинации для второго игрока — второй игрок на четвертом ходе игры независимо от действий первого игрока на третьем ходе получает выигрышную комбинацию;

3) (4,3,8) — невыигрышный ход второго игрока — первый игрок выигрывает на третьем ходе, получив комбинацию (4,3,16).

Следовательно, при ходе первого игрока (4,3,4) второй игрок выигрывает на третьем ходе, для чего на втором ходе игры он должен получить комбинацию (4,6,4) или (6,5,6).

II. Допустим, ход первого игрока — $(2,6,4)$, тогда возможны следующие варианты хода второго игрока:

- 1) $(4,6,4)$ — выигрышная комбинация для второго игрока;
- 2) $(2,12,4)$, $(2,6,8)$ и $(4,8,6)$ — невыигрышные ходы для второго игрока.

Здесь при ходе первого игрока $(2,6,4)$ выигрывает второй игрок, получая комбинацию $(4,6,4)$.

III. Допустим, ход первого игрока — $(2,3,8)$, тогда возможны следующие варианты хода второго игрока:

- 1) $(4,3,8)$, $(2,6,8)$ и $(4,5,10)$ — невыигрышный ход для второго игрока — первый игрок выигрывает на третьем ходе, получив комбинацию $(4,3,16)$;
- 2) $(2,3,16)$ — выигрышный ход для второго игрока.

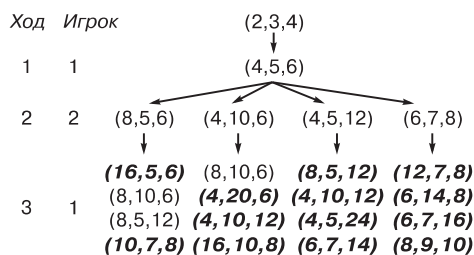
При ходе первого игрока $(2,6,4)$ выигрывает второй игрок, получая комбинацию $(4,6,4)$.

IV. Предположим, что первый игрок своим первым ходом получил комбинацию $(4,5,6)$. Тогда в дальнейшем вне зависимости от хода второго игрока на третьем ходе первый игрок получает выигрышную комбинацию. Таким образом, в соответствии со своей стратегией первый игрок первым своим ходом должен получить комбинацию $(4,5,6)$ и победить в игре на третьем ходе.

Примечание. Можно не строить дерево 2, если сразу ясно, что выигрывает первый игрок.

Ответ:

- 1) выигрывает игрок, который делает первый ход;
- 2) следующее дерево игры доказывает, что выигрывает первый игрок:



- 3) первый ход выигрывающего игрока: первый игрок должен своим первым ходом получить комбинацию $(4,5,6)$, т. е. добавить в каждую из куч по 2 камня.

Задачи для самостоятельного решения

Логические выражения и их преобразование

Задача 1. Определите, удовлетворяет ли имя «Нина» логическому условию (последняя буква гласная \vee первая буква гласная $\wedge \neg$ вторая буква согласная).

Задача 2. Определите, удовлетворяет ли имя «Дмитрий» логическому условию $\neg (\neg (\text{вторая буква согласная} \rightarrow \text{четвертая буква гласная}) \vee (\text{третья буква согласная} \wedge \text{вторая буква согласная}))$.

Задача 3. Определите, при каких значениях переменных P и Q истинно выражение $P \wedge Q \vee P \wedge \neg (P \vee Q)$.

Задача 4. Определите, при каких значениях переменных P и Q истинно выражение $\neg (\neg (P \vee Q) \rightarrow (P \wedge Q))$.

Задача 5. Определите, при каких значениях переменных P и Q ложно выражение $(\neg Q \rightarrow P) \wedge Q \vee P \wedge (Q \rightarrow \neg P)$.

Задача 6. Определите, при каких значениях переменных P и Q ложно выражение $(\neg P \rightarrow \neg Q) \vee P \wedge Q$.

Задача 7. Определите значение высказывания $((X < 5) \wedge (X > 2)) \wedge (X > 3)$ при $X = 2$.

Задача 8. Определите значение высказывания $((X > 1) \rightarrow (X < 4)) \wedge (X < 2)$ при $X = 1$.

Задача 9. Определите значение высказывания $((X < 5) \wedge (X < 3)) \rightarrow \neg (X > 1)$ при $X = 2$.

Задача 10. Определите значение высказывания $((X > 2) \vee (X > 1)) \rightarrow \neg ((X > 5) \vee (X < 4))$ при $X = 4$.

Задача 11. Определите значение высказывания $(\cos(y) < \pi) \wedge (x = y \bmod 3) \vee (y \operatorname{div} x > 2)$ при $x = 1, y = 4$.

Задача 12. Определите значение высказывания $(\sin(x) < \pi) \rightarrow ((x \bmod 5 = y) \vee (y \operatorname{div} x < 1))$ при $x = 3, y = 4$.

Задача 13. Вычислите значение выражения $((x^2 + z^2) \geq y^2)$ И $((x^2 > z^2)$ ИЛИ НЕ $((x + y + z) > 6))$ при $x = 10, y = 1, z = 5$.

Задача 14. Вычислите значение выражения $((\max(y, z) > \min(x, z))$ И $((|y| > |x|)$ ИЛИ $(x^2 > z^2)))$ при $x = 7, y = 3, z = 4$.

Задача 15. Вычислите значение выражения $((((x + y)^2 + 1) > z^2)$ И НЕ $((x^2 + y^2 - z^2) > 0)$ И $(x^2 > 100))$ при $x = 2, y = 3, z = 1$.

Задача 16. Определите наименьшее натуральное число X , при котором истинно высказывание $(X + X \cdot (X + 2) \geq 70) \rightarrow ((X - 2) \cdot (X - 3) < 110)$.

Задача 17. Определите наименьшее натуральное число X , при котором истинно высказывание $(X \cdot (X + 2) < 50) \rightarrow (X \cdot X > 35)$.

Задача 18. Сколько существует целых значений числа X , при которых ложно высказывание $(|X| \geq 5) \vee (|X| < 1)$?

Задача 19. Сколько различных решений имеет уравнение $X \wedge \bar{Y} \wedge \bar{Z} \wedge \bar{U} \wedge (V \vee 1) = 1$, где X, Y, Z, U, V — логические переменные?

Задача 20. Сколько существует целых значений числа X , при которых ложно высказывание $(|X| < 5) \wedge (|X| < 1) \wedge (|X| < 10)$?

Задача 21. Сколько различных решений имеет уравнение $(X \vee 1) \wedge \bar{Y} \wedge \bar{Z} \wedge (\bar{U} \vee 1) \vee (V \vee 1) = 1$, где X, Y, Z, U, V — логические переменные?

Задача 22. Определите наибольшее натуральное число X , при котором истинно высказывание $(X \cdot (X + 1) > 75) \rightarrow (X \cdot X < 65)$.

Задача 23. Сколько различных решений имеет уравнение $((K \leftrightarrow L) \vee M) \rightarrow N = 0$, где K, L, M, N — логические переменные?

Задача 24. Укажите значения логических переменных A, B, C, D , при которых логическое выражение $A \wedge (A \rightarrow B) \vee B \wedge (A \rightarrow B) \vee \bar{C} \vee (A \wedge D)$ ложно.

Задача 25. A, B, C — целые числа, для которых истинно высказывание: $\neg(A \equiv 19) \wedge ((C > A) \rightarrow (2B < A) \wedge ((A < C/2) \rightarrow (A \leq 2B + 4)))$. Чему равно A , если $C = 18$, а $B = 8$?

Задача 26. Определите, является ли корнем логического уравнения $X = F(A, B): A \wedge X \oplus (A \vee B) \wedge \bar{X} = (X \wedge \bar{B} + (\bar{A} \vee B))$ выражение $B \oplus A$.

Задача 27. Определите, является ли корнем логического уравнения $X = F(A, B): (X \wedge (\bar{A} + \bar{B})) \wedge A = ((A \oplus B) \wedge \bar{A} \vee X) \vee (A \wedge B)$ выражение $A \rightarrow B$.

Задача 28. Определите, является ли тавтологией формула $((A \equiv B) \vee B) \oplus A \vee (A \rightarrow B)$.

Задача 29. Три студента — Фролов, Самсонов и Зусман — хотят сдать сессию на отлично. Были высказаны следующие предположения:

1) сдача экзаменов на отлично студентом Самсоновым равносильна тому, что сдаст на отлично Фролов или Зусман;

2) неверно, что сдаст на отлично Самсонов или на отлично сдадут и Фролов, и Зусман;

3) студент Зусман не сдаст экзамены на отлично, и это при том, что если Фролов сдаст на одни пятерки, то и Самсонов сдаст также на отлично.

После сессии оказалось, что только одно из трех этих предположений ложно. Кто сдал экзамены на отлично?

Задача 30. Относительно трех участников соревнований — Мидова, Пышкина и Соколова — были высказаны предположения:

- если Соколов будет в тройке победителей, то и Мидов тоже;
- Мидов и Соколов будут или не будут в призерах соревнований одновременно;
- Мидов не будет призером или среди победителей будут Пышкин и Соколов.

После соревнований оказалось, что одно из этих предположений ложно. Кто из спортсменов был призером?

Задача 31. В финал конкурса вышли четыре мальчика: Никита, Руслан, Сергей и Толя. Девочки решили поделиться своими предположениями об итоговом распределении мест.

Оля: Сережа точно будет вторым, а Толик — четвертым.

Аня: Уверена, что Никита будет первым, а вторым — Руслан.

Кристина: Ерунда. Это Никита будет вторым, а Толик — третьим.

Когда подвели итоги, оказалось, что каждая девочка была права только в одном из своих прогнозов. Какие места заняли Никита, Руслан, Сергей и Толя?

Задача 32. Проверяя дневники, классный руководитель заметил, что у мальчика Ромы исправлены все двойки за неделю, а сделать это могли только три его друга: Максим (М), Андрей (А) и Костя (К), которые задержались на перемену в классе. Они были вызваны к директору, где были спрошены о том, кто подделал оценку.

Андрей: Максим этого не делал, это все Костя красной ручкой!

Костя: Я этого не делал, потому что оценку исправил Максим!

Максим: Ничего я не исправлял! Да и Андрей тоже.

Стало известно, что один из мальчиков сказал чистую правду, второй все соврал, а третий сказал правду только в половине своего ответа. Кто же подделал оценку Роме?

Задача 33. Восемь школьников, остававшихся в классе на перемене, были вызваны к директору: один из них разбил окно в кабинете. На вопрос директора, кто это сделал, были получены следующие ответы.

- 1) *Егор*: «Разбил Андрей!»
- 2) *Света*: «Вика разбила.»
- 3) *Оля*: «Разбила Света.»
- 4) *Миша*: «Это кто-то с улицы!»
- 5) *Надя*: «Да, Оля права...»
- 6) *Коля*: «Это либо Вика, либо Света!»
- 7) *Андрей*: «Ни Вика, ни Света этого не делали.»
- 8) *Вика*: «Андрей не бил!»

Кто разбил окно, если известно, что из восьми высказываний истинно только три?

Задача 34. Восемь школьников, остававшихся в классе на перемене, были вызваны к директору: один из них разбил окно в кабинете. На вопрос директора, кто это сделал, были получены следующие ответы.

- 1) *Соня*: «Это сделал Володя.»
- 2) *Миша*: «Это ложь!»
- 3) *Володя*: «Я разбил!»
- 4) *Аня*: «Это я разбила!»
- 5) *Оля*: «Аня не разбивала!»
- 6) *Рома*: «Разбила либо Соня, либо Оля.»
- 7) *Коля*: «Девочки этого не делали.»
- 8) *Толя*: «Коля разбил!»

Кто разбил окно, если известно, что из восьми высказываний истинно только два?

Задача 35. Маму школьника вызвали в школу. Она точно помнит, что:

- 1) ее вызвали учителя химии, истории, физкультуры и музыки;
- 2) имена учителей — Елизавета Евгеньевна, Лада Львовна, Мария Михайловна и Надежда Николаевна;
- 3) кабинеты (и спортзал) вызвавших ее учителей расположены на первом, втором, третьем и четвертом этажах;
- 4) спортзал расположен на первом этаже;
- 5) кабинет химии расположен выше кабинета музыки;
- 6) кабинет истории расположен выше третьего этажа;
- 7) Елизавета Евгеньевна — не химик и не историк;
- 8) Лада Львовна — либо учитель музыки, либо учитель химии;
- 9) кабинет Марии Михайловны ниже третьего этажа;
- 10) физкультуру ведет не Елизавета Евгеньевна.

Помогите маме установить, какого учителя как зовут.

Задача 36. В квартире живут 4 собаки разных пород. В коридоре лежат поводки (черный, красный, синий и зеленый) и ошейники (черный, красный, синий и зеленый). Породы собак — такса, бульдог, ротвейлер и овчарка. Известно, что:

- 1) черный ошейник принадлежит не бульдогу и не овчарке;
- 2) красный ошейник принадлежит либо бульдогу, либо ротвейлеру;
- 3) черные и красные вещи составляют комплекты и принадлежат одним и тем же собакам, остальные вещи комплектов не составляют;
- 4) синий ошейник принадлежит либо ротвейлеру, либо овчарке;
- 5) зеленый ошейник — либо у таксы, либо у овчарки;
- 6) ротвейлер не является обладателем комплекта.

Определите цвет ошейника каждой собаки.

Задача 37. Четыре рядом стоящих дома расположены по одной стороне улицы. В них живут Маша, Света, Ира и Валя. Из каждого дома сбежало по кошке: черная, серая, белая и трехцветная. Нашедший всех четырех животных точно знает, что:

- 1) Валя — соседка Маши;
- 2) Ира живет левее Вали и Маши;
- 3) Света живет правее и Вали, и Маши;
- 4) Света не соседка Вали;
- 5) у Иры кошка не черная и не серая;
- 6) белая кошка не живет в крайнем доме;
- 7) Маша — хозяйка черной кошки.

Определите цвет кошек, принадлежащих каждой девочке.

Построение таблиц истинности логических выражений

Задача 1. Составьте таблицу истинности для функции $F = \overline{(A \oplus C)} \vee \overline{A} \wedge \overline{C} \vee B$, в которой столбец значений аргумента A представляет собой двоичную запись числа 15, столбец значений аргумента B — запись числа 51, а столбец значений аргумента C — запись числа 85. Число в столбце записывается сверху вниз от старшего разряда к младшему. Переведите полученную двоичную запись значения функции F в десятичную систему счисления.

Задача 2. Составьте таблицу истинности для функции $F = \overline{A} \wedge \overline{C} \vee \overline{A} \vee (B \rightarrow C)$, в которой столбец значений аргумента A представляет собой двоичную запись числа 15, столбец значений аргумента B — запись числа 51, а столбец значений аргумента C — запись числа 85. Число в столбце записывается сверху вниз от старшего разряда к младшему. Переведите полученную двоичную запись значения функции F в десятичную систему счисления.

Задача 3. Определите, является ли тождественно ложной (противоречивой) формула $(A \rightarrow B) \wedge A \rightarrow (B \vee C)$.

Задача 4. Определите, является ли тождественно ложной (противоречивой) формула $A \vee B \vee (A \rightarrow C) \rightarrow \bar{B}$.

Задача 5. Символом F обозначено логическое выражение $(Y \wedge X \vee Z) \rightarrow (\neg X \vee \neg Z)$ от трех аргументов: X, Y, Z . Дан фрагмент таблицы истинности выражения F :

X	Y	Z	F
0	1	0	1
0	1	1	1
1	0	0	1

Определите, соответствует ли данное выражение значению F .

Задача 6. Символом F обозначено логическое выражение $(X \rightarrow Y) \rightarrow Z \wedge \neg Y$ от трех аргументов: X, Y, Z . Дан фрагмент таблицы истинности выражения F :

X	Y	Z	F
0	1	1	1
1	0	0	1
1	1	1	1

Определите, соответствует ли данное выражение значению F .

Задача 7. Символом F обозначено логическое выражение $(X \rightarrow \neg Y) \wedge (Y \rightarrow \neg Z) \wedge (Z \rightarrow \neg X)$ от трех аргументов: X, Y, Z . Дан фрагмент таблицы истинности выражения F :

X	Y	Z	F
0	0	0	1
0	1	0	1
1	1	0	0

Определите, соответствует ли данное выражение значению F .

Задача 8. Дана таблица истинности выражения F :

X	Y	Z	F
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

вает игрок, после хода которого расстояние по прямой от фигуры до точки с координатами $(0,0)$ больше 10 единиц. Кто выигрывает при безошибочной игре обоих игроков — игрок, делающий первый ход, или игрок, делающий второй ход? Каким должен быть первый ход выигрывающего игрока?

Задача 2. Два игрока играют в следующую игру. Перед ними лежат две кучки камней, в первой из которых 2, а во второй — 3 камня. У каждого игрока неограниченно много камней. Игроки ходят по очереди. Ход состоит в том, что игрок или увеличивает в 2 раза количество камней в какой-то куче, или добавляет 3 камня в какую-то кучу. Выигрывает игрок, после хода которого общее число камней в двух кучах становится не менее 18. Кто выигрывает при безошибочной игре обоих игроков — игрок, делающий первый ход, или игрок, делающий второй ход? Каким должен быть первый ход выигрывающего игрока?

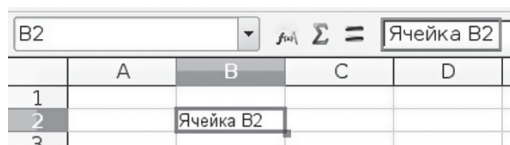
Задача 3. Даны три кучи камней, содержащих соответственно 1, 3 и 5 камней. За один ход разрешается или удвоить количество камней в меньшей куче (если их две — то в каждой из них), или добавить по 1 камню в каждую из всех трех куч. Выигрывает игрок, после хода которого во всех трех кучах суммарно становится не менее 16 камней. Игроки ходят по очереди. Выясните, кто выигрывает при правильной игре, — первый или второй игрок.

Глава 3

Средства ИКТ

3.1. Обработка информации в электронных таблицах

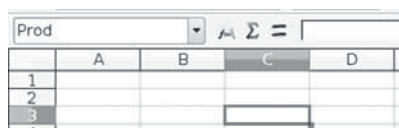
Электронная таблица (табличный процессор) — прикладная программа, предназначенная для обработки данных, представленных в форме таблицы. Электронная таблица представляет собой компьютерный вариант так называемой рабочей книги для проведения расчетов. Такая книга состоит из отдельных именованных *листов* (таблиц), которые формируются из *ячеек*. Столбцы таблицы именуются заглавными латинскими буквами (A, B, ..., Z, AA, ..., AZ, ...), а строки — числами. При этом каждая ячейка имеет уникальный *адрес (имя)*, составленный из имени столбца и номера строки, на пересечении которых она находится, — например, ячейка с адресом **B2** находится на пересечении второй строки и столбца B (рис. 3.1).



	A	B	C	D
1				
2		Ячейка B2		
3				

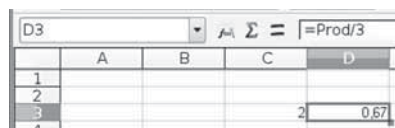
Рис. 3.1. Адрес ячейки

Ячейке (либо диапазону ячеек) можно назначить *имя* (рис. 3.2). Оно используется в формулах как замена абсолютного адреса ячейки. Например, назначив ячейке C3 имя «Prod», в ячейку D3 можно поместить формулу: **=Prod/3** (вместо формулы **=C3/3**). В этом случае при копировании формулы адрес ячейки не будет меняться.



	A	B	C	D
1				
2				
3			Ячейка C3	

а) Присвоение имени ячейки



	A	B	C	D
1				
2				
3				0.67

б) Использование имени ячейки

Рис. 3.2. Работа с ячейкой

Содержимым ячейки может быть:

1) число — целое со знаком или без знака (например, -345), дробное с фиксированной запятой (253,62) или с плавающей запятой (2,5362E+2);

2) текст;

3) формула, в соответствии с которой вычисляется значение, хранящееся в ячейке. Запись формулы начинается с символа «=» (знак равенства).

Любая информация, которая не может быть истолкована как формула или число, считается текстом.

По умолчанию числа в ячейках электронной таблицы выравниваются по правому краю ячейки, а текст — по левому.

Во многих странах для отделения целой части числа от дробной используется символ «.» (точка), в то время как в России — символ «,» (запятая). Введя в одну ячейку текст, содержащий число, но с различными разделителями, можно определить символ, который используется для отделения дробной части.

Формулы могут содержать ссылки на данные, размещенные в других ячейках. Например, если ячейка **C1** содержит формулу **=A1+B1**, то в ней будет отображаться сумма значений, расположенных в ячейках **A1** и **B1**, причем при изменении данных в этих ячейках будет выполняться автоматический пересчет этой суммы. Копирование формул из одной ячейки в другие позволяет значительно ускорить процесс подготовки таблицы, содержащей однотипные формулы.

В формулах можно использовать числовые константы, ссылки на отдельные ячейки и блоки ячеек, знаки арифметических операций и встроенные функции (**СУММ**, **SIN** и др.) — математические, статистические, финансовые и др.

Операция	Обозначение в формуле	Пример
Возведение в степень	\wedge	=3 ²
Умножение	*	=A1*B2
Деление	/	=B1/B2
Сложение	+	=A1+A2
Вычитание	–	=A1-A2
Обозначение диапазона ячеек	:	=СУММ(A1:B3)
Объединение нескольких диапазонов ячеек	;	=СУММ(A1:B3;C3)
<i>Операции сравнения чисел</i>		
Проверка на равенство	=	=B1=B2
Меньше	<	=B1<B2
Больше	>	=B1>B2
Меньше или равно	<=	=B1<=B2
Больше или равно	>=	=B1>=B2
Не равно	<>	=B1<>B2

Электронная таблица имеет два режима отображения: *отображение данных* (результатов вычислений) и *отображение формул*, по которым выполняется расчет данных (рис. 3.3).

A1 Σ = =12+B1				
	A	B	C	D
1	=12+B1	1		
2				
3				

а) Отображение формул

A1 Σ = =12+B1				
	A	B	C	D
1	13	1		
2				
3				

б) Отображение данных

Рис. 3.3. Режимы отображения

3.1.1. Автозаполнение

Автозаполнение — функция электронной таблицы, которая позволяет на основе значения, введенного в исходную ячейку, задать значения для ячеек выделенного диапазона. Например, если ввести в одну из ячеек слово «Январь» и выделить *маркером автозаполнения* несколько ячеек, то функция автозаполнения поместит в эти ячейки слова «Февраль», «Март» и т. д. Данная функция основана на ранее созданных *списках последовательностей* (списках автозаполнения). Например, такая последовательность определена для названий месяцев (Январь, Февраль, ..., Ноябрь, Декабрь). Поэтому при попытке применить эту функцию к содержимому ячейки, для которой список автозаполнения не создан, выделенный диапазон будет заполнен копиями содержимого исходной ячейки. Например, если ввести в ячейку некоторое слово (скажем, «Пример»), а затем выделить с помощью маркера автозаполнения некоторый диапазон, то это слово будет повторено во всех ячейках выделенного диапазона, так как в автозаполнении не определена последовательность, содержащая слово «Пример». Однако вы можете сами создать любую последовательность, которая затем будет использоваться функцией автозаполнения.

Автозаполнение выполняется в зависимости от содержимого ячейки, к которой применяется данное действие.

Содержимое ячейки	Выделение ячеек маркером автозаполнения	Выделение ячеек при нажатой клавише Ctrl
Элемент списка автозаполнения («Январь»)	Будет продолжен ряд («Январь», «Февраль», «Март», «Апрель», «Май» ...)	Диапазон заполняется копиями элемента («Январь», «Январь», «Январь», «Январь» ...)

Содержимое ячейки	Выделение ячеек маркером автозаполнения	Выделение ячеек при нажатой клавише Ctrl
Комбинация текста и числа («ном.1»)	Числовое значение будет наращиваться («ном.1», «ном.2», «ном.3», «ном.4», «ном.5»...)	Диапазон заполняется копиями значения исходной ячейки («ном.1», «ном.1», «ном.1», «ном.1» ...)
Число (в исходной ячейке находится число 1)	В каждую ячейку выделенного диапазона помещается число, находящееся в исходной ячейке (1, 1, 1, 1, 1 ...)	Диапазон заполняется арифметической прогрессией с шагом 1 (1, 2, 3, 4, 5 ...)
Текст (слово «дата»)	Каждая ячейка выделенного диапазона будет заполнена текстом исходной ячейки («дата», «дата», «дата», «дата», «дата» ...)	
Формула	Диапазон заполнится копиями формулы, находящейся в исходной ячейке	

Если требуется наращивать числовое значение в комбинированном элементе данных (содержащем текст и число), то необходимо, чтобы число было отделено от текста пробелом или находилось после текста. Если чисел в комбинированном элементе данных содержится несколько, то наращиваться будет первое из них.

Выше описаны особенности работы функции автозаполнения в Microsoft Excel. В других электронных таблицах поведение этой функции при заполнении ячеек, возможно, будет отличаться от описанного.

3.1.2. Относительная и абсолютная адресация

Одна из возможностей электронных таблиц заключается в использовании в формулах ссылок на другие ячейки (адресов ячеек). При вычислении по формуле вместо адреса указанной ячейки подставляется ее текущее значение (число, текст и т. д.).

При указании ячейки используют ее относительный или абсолютный адрес. Признаком *абсолютной адресации* является символ \$ перед номером столбца и/или строки, например: \$A\$1, B\$5, \$C16 (т. е. координата строки и координата столбца в адресе ячейки могут фиксироваться как абсолютные отдельно друг от друга).

Адресация		По строке	
		Относительная	Абсолютная
По столбцу	Относительная	D1	D\$1
	Абсолютная	\$D1	\$D\$1

Рассмотрим, как будут изменяться адреса ячеек в формуле вида $=\$A\$1+B1$ при ее копировании из ячейки **B3** в ячейку **C4**. В исходной формуле адрес $\$A\1 является абсолютным, следовательно, при копировании в любую ячейку он изменяться не будет.

Формула в B3	Формула в C4	Примечание
$=\$A\$1+B1$	$=\$A\$1+C2$	Адрес B2 — относительный, поэтому при копировании изменится как имя столбца, так и номер строки (рис. 3.4, а)
$=\$A\$1+B\$1$	$=\$A\$1+C\$1$	При копировании в адресе B\$1 изменится имя столбца, а номер строки останется неизменным (рис. 3.4, б)
$=\$A\$1+$B1$	$=\$A\$1+$B2$	При копировании в адресе \$B1 изменится номер строки, а имя столбца останется неизменным (рис. 3.4, в)
$=\$A\$1+$B1	$=\$A\$1+$B1	Адрес \$B\$1 — абсолютный, поэтому при копировании он не изменится (рис. 3.4, г)

	A	B	C	D
1				
2				
3		$=\$A\$1+B1$		
4			$=\$A\$1+C2$	
5				
6				

а)

	A	B	C	D
1				
2				
3		$=\$A\$1+B\$1$		
4			$=\$A\$1+C\$1$	
5				
6				

б)

	A	B	C	D
1				
2				
3		$=\$A\$1+$B1$		
4			$=\$A\$1+$B2$	
5				
6				

в)

	A	B	C	D
1				
2				
3		$=\$A\$1+$B\1		
4			$=\$A\$1+$B\1	
5				
6				

г)

Рис. 3.4. Изменение адресов в формулах при копировании

3.1.3. Функции электронных таблиц

Функции — это специальные, заранее созданные формулы, которые позволяют выполнять различного рода расчеты — вычисление квадратного корня числа, среднего набора чисел и т. п. Рассмотрим возможности использования функций на примере Microsoft Excel.

Для вызова функции необходимо указать имя функции и передаваемые ей значения аргументов. Например, в формуле

=СУММ(B1:B3): запись **СУММ** — это имя функции, а **B1:B5** — аргумент (диапазон ячеек). Данная формула суммирует числа, находящиеся в ячейках **B1**, **B2** и **B3**.

Список аргументов функции должен быть заключен в круглые скобки. Открывающая скобка отмечает начало списка аргументов и ставится сразу после имени функции (при наличии пробела или другого символа между именем и открывающей скобкой в ячейке будет отображено сообщение об ошибке: **#ИМЯ?**). Некоторые функции не имеют аргументов, но даже в этом случае функция должна содержать круглые скобки, например: **=С5*ПИ()** (эта формула выполняет умножение значения, хранящегося в ячейке **С5**, на величину π ($\approx 3,14$)).

При использовании в функции нескольких аргументов они отделяются друг от друга точкой с запятой. Например, следующая формула указывает, что необходимо перемножить числа в ячейках **A1**, **A3** и **A6**: **=ПРОИЗВЕД(A1;A3;A6)**.

Допустимые аргументы функции

- 1) диапазон ячеек — пример: **=СУММ(A2:A5;B4:B8)**;
- 2) число — **=СУММ(24;987;49)**;
- 3) текст — (**=ТЕКСТ(ТДАТА0;"Д.ММ.ГГ")**) — выводит в ячейке значение «12.01.10».

В последней формуле второй аргумент функции **ТЕКСТ** является текстовым и задает шаблон для преобразования десятичного значения даты, возвращаемого функцией **ТДАТА0**, в строку символов. Текстовый аргумент может представлять собой строку символов, заключенную в двойные кавычки, или ссылку на ячейку, которая содержит текст.

В качестве аргумента функции можно указать *имя диапазона*. Например, если диапазону ячеек **A1:A5** присвоено имя **Диапазон1**, то для вычисления суммы чисел в ячейках с **A1** по **A5** можно использовать формулу **=СУММ(Диапазон1)**.

В одной и той же функции можно использовать аргументы различных типов, например: **=СРЗНАЧ(Диапазон1; С5;2*8)**.

Аргументы некоторых функций могут принимать только логические значения **ИСТИНА** или **ЛОЖЬ**. Логическое выражение возвращает значение **ИСТИНА** или **ЛОЖЬ** в ячейку или формулу, содержащую это выражение.

В Microsoft Excel для работы с логическими величинами используются функции, представленные в таблице:

Функция	Описание и формат записи	Пример использования
И	И(лог_значение1; лог_значение2; ...) возвращает значение ИСТИНА , если все аргументы имеют значение ИСТИНА	=И(2>3; 4<5) — возвращает значение ЛОЖЬ , так как первый аргумент функции — ЛОЖЬ
ИЛИ	ИЛИ(лог_значение1; лог_значение2; ...) возвращает значение ИСТИНА , если хотя бы один аргумент имеет значение ИСТИНА	=ИЛИ(2>3; 4<5) — возвращает значение ИСТИНА , так как второй аргумент функции — ИСТИНА
НЕ	НЕ(лог_значение) меняет логическое значение своего аргумента на противоположное	=НЕ(2>3) — возвращает значение ИСТИНА
ИСТИНА	ИСТИНА() возвращает логическое значение ИСТИНА	= ИСТИНА()
ЛОЖЬ	ЛОЖЬ() возвращает логическое значение ЛОЖЬ	=ЛОЖЬ()
ЕСЛИ	ЕСЛИ(лог_выражение; значение_если_истина; значение_если_ложь) выполняет проверку условия: <i>лог_выражение</i> — выражение, значение которого вычисляется; <i>значение_если_истина</i> — действия, которые будут выполняться, если результат вычисления <i>лог_выражения</i> — ИСТИНА ; <i>значение_если_ложь</i> — если ЛОЖЬ	=ЕСЛИ(2>3; "больше"; "меньше")

Задача 1. В электронной таблице значение формулы **=СУММ(В1:В2)** равно 5. Чему равно значение ячейки **В3**, если значение формулы **=СРЗНАЧ(В1:В3)** равно 3 [6]?

Решение.

Высказывание «**=СУММ(В1:В2)** равно 5» означает, что сумма чисел в ячейках **В1** и **В2** равна 5. Высказывание «**=СРЗНАЧ(В1:В3)** равно 3» указывает на то, что среднее значение содержимого ячеек **В1**, **В2** и **В3** равно 3. Учитывая это, можно записать формулы:

$$\mathbf{B1 + B2 = 5,} \quad (3.1)$$

$$\mathbf{(B1 + B2 + B3) / 3 = 3.} \quad (3.2)$$

Подставляя (3.1) в (3.2), получим:

$$(5 + B3) / 3 = 3 \Rightarrow 5 + B3 = 9 \Rightarrow B3 = 4.$$

Таким образом, в ячейке **B3** хранится число 4.

Ответ: 4.

Задача 2. В ячейке **C2** записана формула **\$E\$3+D2**. Какой вид приобретет эта формула после того, как ячейку **C2** скопируют в ячейку **B1**? Примечание: знак \$ используется для обозначения абсолютной адресации [2].

Решение.

Рассмотрим слагаемые указанной формулы:

1) для ячейки **\$E\$3** записан абсолютный адрес, который при копировании формулы в другую ячейку меняться не будет;

2) адрес ячейки **D2** – относительный, поэтому данная часть формулы при копировании в другую ячейку изменится.

При копировании формулы из ячейки **C2** в ячейку **B1** номер строки уменьшается на 1, поэтому номер строки в относительном адресе также изменяется на одну единицу «назад» (в сторону уменьшения). Аналогично — с именем столбца.

Следовательно, после копирования формулы в адресе ячейки **D2** номер строки уменьшится на единицу, а вместо имени столбца будет записана предыдущая буква, т. е. из **D2** получится адрес **C1**.

Ответ: **\$E\$3 + C1**.

Задача 3. Дан фрагмент электронной таблицы:

	A	B	C
1	4	2	
2	1	3	=A2 + 2*B\$2

Чему станет равно значение ячейки **C1**, если в нее скопировать формулу из ячейки **C2**? Знак \$ обозначает абсолютную адресацию.

Решение.

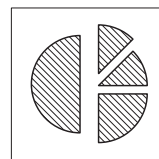
При копировании формулы из ячейки **C2** в **C1** в адресах ячеек, используемых в этой формуле, изменится только номер строки — он уменьшится на единицу.

Рассмотрим формулу **=A2 + 2*B\$2**. Адрес ячейки **A2** — относительный, следовательно, при копировании формулы в ячейку **C1** в нем изменится номер строки: **A2** преобразуется в **A1** (номер строки уменьшится на единицу). В адресе же ячейки **B\$2** номер строки защищен от изменения (он — абсолютный).

Учитывая это, в ячейке **C1** мы получим формулу: **=A1 + 2*B\$2**.
 Вычислим значение ячейки **C1**, используя указанную формулу:
 $4 + 2 \cdot 3 = 10$.
Ответ: 10.

Задача 4. Дан фрагмент электронной таблицы

	A	B	C	D
1		3	4	
2	=C1-B1	=B1-A2*2	=C1/2	=B1+B2



После выполнения вычислений была построена диаграмма по значениям диапазона ячеек **A2:D2**. Определите, соответствует ли приведенная диаграмма данным этого диапазона [5].

Решение.

Вычислим значения ячеек **A2:D2**, используя таблицу, приведенную в условии задачи:

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

Таким образом, диаграмма должна состоять из четырех элементов, два из которых составляют $1/8$ от всей диаграммы, один — $2/8$ и еще один — $4/8$.

Приведенная в условии диаграмма тоже состоит из четырех элементов: размером $1/8$, $2/8$ и $4/8$ от всей диаграммы. Следовательно, данная диаграмма построена по значениям диапазона ячеек **A2:D2**.

Ответ: соответствует.

Задача 5. В цехе трудятся рабочие трех специальностей — токари (Т), слесари (С) и фрезеровщики (Ф). Каждый рабочий имеет разряд, не меньший второго и не больший пятого. На диаграмме 1 показано распределение рабочих по специальностям, а на диаграмме 2 — количество рабочих с различными разрядами. Каждый рабочий может иметь только одну специальность и только один разряд.

Имеется четыре утверждения:

- а) среди слесарей найдется хотя бы один третьего разряда;
- б) среди токарей найдется хотя бы один второго разряда;
- в) все токари могут иметь четвертый разряд;
- г) все фрезеровщики могут иметь третий разряд.

Диаграмма 1

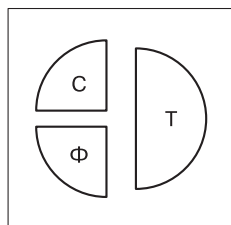
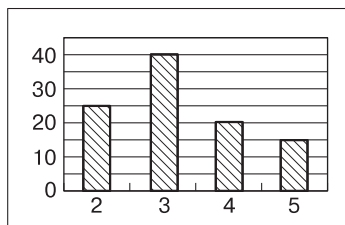


Диаграмма 2



Какое из этих утверждений следует из анализа обеих диаграмм [3]?

Решение.

Из первой диаграммы мы получаем, что слесари составляют 25% от общего числа рабочих, фрезеровщики — 25%, а токари — 50%.

Из второй диаграммы получаем, что в цехе трудятся 25 рабочих второго разряда, 40 — третьего разряда, 20 — четвертого разряда и 15 — пятого разряда, т. е. всего $25 + 40 + 20 + 15 = 100$ человек.

Рассмотрим каждое утверждение:

а) среди слесарей найдется хотя бы один третьего разряда — данное утверждение не следует из обеих диаграмм, так как неизвестно, какое количество слесарей имеет третий разряд;

б) среди токарей найдется хотя бы один второго разряда — данное утверждение не следует из обеих диаграмм (рассуждения аналогичны предыдущим);

в) все токари могут иметь четвертый разряд — токарей всего 50 человек, а рабочих четвертого разряда — 20 человек, следовательно, все токари не могут быть четвертого разряда;

г) все фрезеровщики могут иметь третий разряд — всего фрезеровщиков 25 человек, а рабочих третьего разряда — 40 человек, следовательно, все фрезеровщики могут быть третьего разряда.

Ответ: г.

Задача 6. Дан фрагмент электронной таблицы

	A	B	C	D
1		Курс \$	32	
2	Товар	Цена (\$)	Количество	Стоимость (руб.)
3	Монитор	200	4	
4	Клавиатура	10	10	
5	CD-ROM	30	5	

Какая формула должна быть введена в ячейку **D3** для последующего автозаполнения нижестоящих значений?

Решение.

Для подсчета стоимости товара в рублях необходимо перемножить количество товара, цену за единицу товара в долларах и курс доллара. Для каждого товара ячейка, в которой хранится цена и количество, своя (т. е. для этих ячеек в формулах должны использоваться относительные адреса), а курс доллара — общий для всех формул (т. е. должен использоваться абсолютный адрес).

Запишем формулу для вычисления значения в ячейке **D3**, используя указанные замечания: $=\$C\$1*B3*C3$.

Ответ: $=\$C\$1*B3*C3$.

Задача 7. Представлен фрагмент электронной таблицы, содержащий числа и формулы.

	B	C	D
1	1	2	
2	3	4	$=\text{СЧЕТ}(\text{B1:C2})$
3			$=\text{СРЗНАЧ}(\text{B1:D2})$

На сколько по абсолютной величине изменится значение в ячейке **D3** после перемещения содержимого ячейки **C2** в ячейку **C3**?

Решение.

Функция **СРЗНАЧ** вычисляет среднее значение содержимого ячеек, которые указаны в качестве ее аргументов.

Функция **СЧЕТ** подсчитывает количество чисел в списке аргументов, т. е. количество числовых ячеек в указанном интервале.

Рассмотрим два варианта указанной таблицы до и после *перемещения* содержимого ячейки **C2** в ячейку **C3** (операция перемещения заключается в том, что содержимое ячейки **C2** вырезается и вставляется в ячейку **C3**). Вычислим также значения ячеек по формулам, приведенным в таблице в условии задачи. (Необходимо помнить, что после перемещения значения ячеек, в которых записаны формулы $=\text{СЧЕТ}(\text{B1:C2})$ и $=\text{СРЗНАЧ}(\text{B1:D2})$ обновляются.)

До перемещения

	B	C	D
1	1	2	
2	3	4	4
3			$2.8 = (1 + 2 + 3 + 4 + 4) / 5$

После перемещения

	B	C	D
69	1	2	
70	3		3
71		4	$2.25 = (1 + 2 + 3 + 3) / 4$

После анализа значений ячеек **D3** и **D71** в таблице до и после перемещения можно отметить, что значение в ячейке **D71** изменится по абсолютной величине на 0.55.

Ответ: 0.55.

Задача 8. Дан фрагмент электронной таблицы в режиме отображения формул:

	A	B	C	D	E
1	2	6			
2	5	1	2	3	4
3	=A2+\$A\$1	=A3*B2	=-C2+2*\$B\$1	=D2+A3*2	=E2+\$B\$1

Определите сумму значений диапазона **C2:C6** после копирования диапазона ячеек **A3:E3** в диапазон **A4:E6**.

Решение.

Проанализируем исходные данные. Значение ячейки **C3** рассчитывается по значению ячеек **C2** и **\$B\$1**. После копирования диапазона ячеек **A3:E3** в диапазон **A4:E6** формула, находящаяся в ячейке **C3**, будет скопирована в ячейки **C4:C6**. В результате в этих ячейках будут находиться формулы:

$$C4 = -C3 * \$B\$1;$$

$$C5 = -C4 * \$B\$1;$$

$$C6 = -C5 * \$B\$1.$$

Таким образом, значение в каждой ячейке диапазона **C4:C6** рассчитывается на основе значения предыдущей ячейки, находящейся в том же столбце, и значения ячейки **\$B\$1**. Поэтому значения в **C3:C6** будут зависеть только от предыдущих значений и не будут зависеть от значений других столбцов. Из этого следует, что для определения суммы требуется вычислить только значения ячеек **C3:C6**:

$$C3 = -1 + 2 * 5 = 9;$$

$$C4 = -9 + 2 * 5 = 1;$$

$$C5 = -1 + 2 * 5 = 9;$$

$$C6 = -9 + 2 * 5 = 1.$$

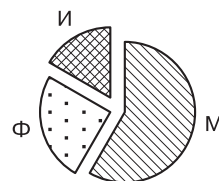
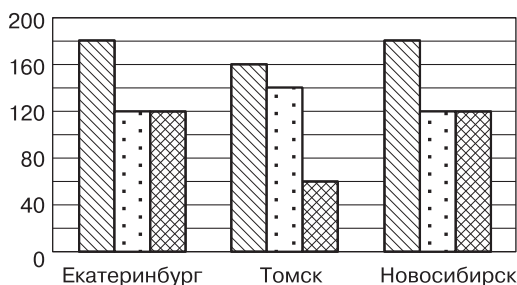
В результате мы получим таблицу:

	A	B	C	D	E
1	2	6			
2	5	1	2	3	4
3			10		
4			2		
5			10		
6			2		

Сумма значений диапазона ячеек **C2:C6** равна 26.

Ответ: 26.

Задача 9. На диаграмме (слева) показано количество призеров по информатике (**И**), математике (**М**) и физике (**Ф**) в трех городах России.



Определите, верно ли отражает приведенная диаграмма (справа) соотношение призеров из всех городов по каждому предмету [6].

Решение.

Вычислим количество призеров по каждому предмету по всем городам:

- 1) математика — $180 + 160 + 180 = 520$ призеров;
- 2) физика — $120 + 140 + 120 = 380$ призеров;
- 3) информатика — $120 + 60 + 120 = 300$ призеров.

Проанализируем полученную информацию относительно количества призеров:

- 1) диаграмма должна состоять из трех частей различного размера;
- 2) на диаграмме количество призеров по физике должно быть больше, чем призеров по информатике;
- 3) в сумме количества призеров по физике и информатике не составляют половину от всех призеров.

Приведенная диаграмма отвечает всем этим условиям.

Ответ: верно.

3.2. Организация баз данных

По определению К. Дейта, **база данных** — это некоторый набор постоянно хранимых данных, используемых прикладными программными системами какого-либо предприятия. Для создания, редактирования и работы с базой данных используется *система управления базой данных (СУБД, DBMS, DataBase Management System)*.

Существуют различные *модели представления данных*: иерархическая, сетевая, реляционная, объектно-ориентированная, объектно-реляционная.

В настоящее время для решения задач организации данных часто применяется *реляционная модель данных*. В ней основной объект базы данных — *таблица*. Каждая таблица включает информацию об объектах определенного типа, например о товарах, поставщиках, клиентах и т. д. Таблица состоит из *полей* (столбцов) и *записей* (строк). *Запись* — это строка, содержащая в себе полный набор данных об описываемом объекте. Записи в таблице отличаются только содержимым их полей. Две записи, в которых все поля одинаковы, считаются идентичными.

Ниже представлен фрагмент таблицы базы данных телефонных абонентов:

		Поля			
Записи	№ записи	Фамилия	Имя	Отчество	№ телефона
	1	Петров	Сергей	Иванович	6546544
	2	Иванов	Алексей	Петрович	5894565
	3	Сидорова	Мария	Ивановна	2237898
	4	Алешина	Мария	Петровна	3214568

Каждая таблица имеет собственный, заранее определенный набор именованных колонок (полей). Поля таблицы обычно соответствуют атрибутам объектов, которые необходимо хранить в базе. Количество строк (записей) в таблице неограниченно, и каждая запись соответствует отдельному объекту.

При создании таблицы для каждого ее поля указывается имя, тип и размер. Чаще всего используются следующие типы данных:

1) *текст* — текст или комбинация текста и чисел (например, адреса, номера телефонов, инвентарные номера или почтовые индексы);

2) *число* — используется для хранения чисел;

3) *дата/время* — используется для хранения значений дат и времени;

4) *денежный* — используется для денежных значений и для предотвращения ненужного округления во время вычислений;

5) *логический* — данные принимают только одно из двух возможных значений, таких как «Да/Нет» или «Истина/Ложь».

Задача 1. Представлена таблица базы данных «Классы школы»:

Класс	Кол_учеников	Староста
9а	20	Иванов
10а	25	Петров
8в	18	Чаадаев
10б	19	Сидоров
7б	21	Лукин

Куда переместятся сведения об Иванове после сортировки таблицы в порядке возрастания по полю «Класс»?

Решение.

В поле «Класс» хранится строка символов. При сортировке такие строки будут сравниваться по кодам символов, составляющих строку. (В кодировке CP1251 русские буквы располагаются по алфавиту, а коды цифр 0–9 располагаются по возрастанию значений. При сравнении строк разной длины считается, что строка с меньшей длиной дополняется символами с нулевым кодом.)

Код символа «9» больше кодов символов «1» и «8», поэтому в результате сортировки в последнюю строку будет записана информация о девятом классе:

Класс	Кол_учеников	Староста
9а	20	Иванов

Следующая строка, содержащая информацию о восьмом классе, будет расположена над строкой с информацией о девятом классе:

Класс	Кол_учеников	Староста
8в	18	Чаадаев
9а	20	Иванов

Далее в таблицу будет помещена строка, включающая информацию о седьмом классе:

Класс	Кол_учеников	Староста
7б	21	Лукин
8в	18	Чаадаев
9а	20	Иванов

Сравним строки «10б» и «10а». Обе эти строки начинаются одинаково, следовательно, все зависит от третьих символов строк. Код символа «а» меньше кода символа «б», поэтому строка «10б» больше строки «10а».

В результате мы получаем таблицу:

Класс	Кол_учеников	Староста
10а	25	Петров
10б	19	Сидоров
7б	21	Лукин
8в	18	Чаадаев
9а	20	Иванов

Таким образом, сведения об Иванове переместились на четыре строки вниз.

Ответ: сведения об Иванове переместились на четыре строки вниз.

Задача 2. Ниже приведены фрагменты таблиц базы данных участников конкурса исполнительского мастерства:

Страна	Участник	Участник	Инструмент	Автор произведения
Германия	Силин	Альбрехт	флейта	Моцарт
США	Клеменс	Бергер	скрипка	Паганини
Россия	Холево	Каладзе	скрипка	Паганини
Грузия	Яшвили	Клеменс	фортепиано	Бах
Германия	Бергер	Силин	скрипка	Моцарт
Украина	Численко	Феер	флейта	Бах
Германия	Феер	Холево	скрипка	Моцарт
Россия	Каладзе	Численко	фортепиано	Моцарт
Германия	Альбрехт	Яшвили	флейта	Моцарт

Определите, представители скольких стран исполняют Моцарта [7]?

Решение.

Определим участников, которые исполняют Моцарта. К ним относятся: Альбрехт, Силин, Холево, Численко и Яшвили.

Определим страны, которые представляют перечисленные выше участники: Альбрехт — Германия, Силин — Германия, Холево — Россия, Численко — Украина, Яшвили — Грузия.

Таким образом, Моцарта исполняют представители 4 стран.

Ответ: 4.

Задача 3. Представлена база данных «Расписание уроков»:

№	День	№ урока	Кабинет	Предмет	Учитель	Класс
1	пт	2	32	химия	Сидорова	9а
2	пн	4	21	физика	Иванова	8в
3	ср	4	11	инф	Петрова	8б
4	чт	3	32	литер	Зайцев	8а
5	вт	4	31	инф	Петрова	10б
6	пт	3	11	матем	Голубев	8а
7	пн	2	41	химия	Петрова	9а
8	пн	1	14	инф	Петров	11а
9	вт	1	41	химия	Панина	11б

Определите записи, которые удовлетворяют условию отбора:
N_урока > 2 И Класс > "8б".

Решение.

Требуется найти записи, которые удовлетворяют двум условиям:

- 1) **N_урока > 2;**
- 2) **Класс > "8б".**

Первому условию отвечают записи:

№	День	№ урока	Кабинет	Предмет	Учитель	Класс
2	пн	4	21	физика	Иванова	8в
3	ср	4	11	инф	Петрова	8б
4	чт	3	32	литер	Зайцев	8а
5	вт	4	31	инф	Петрова	10б
6	пт	3	11	матем	Голубев	8а

Из приведенных записей выберем те, которые отвечают условию **Класс > "8а"**. Тип поля «Класс» — символьная строка. В этом случае сравнение строк выполняется по кодам символов, составляющих строки, и при сравнении строк разной длины считается, что строка с меньшей длиной дописывается символами с нулевым кодом.

1. **"8в" > "8б"** — строки начинаются с одинаковых символов («8»), но код символа «в» больше кода символа «б», поэтому вторая запись отвечает указанному условию.

2. **"8б" > "8б"** — строки начинаются с одинаковых символов («8»), но код символа «б» не больше кода символа «б», поэтому третья запись не отвечает указанному условию.

3. **"8а" > "8б"** — строки начинаются с одинаковых символов («8»), но код символа «а» не больше кода символа «б», поэтому третья запись не отвечает указанному условию.

4. "106" > "86" — код символа «1» меньше кода символа «8», следовательно, данное условие не выполняется.

5. "8a" > "86" — строки начинаются с одинаковых символов («8»), но код символа «a» не больше кода символа «6», поэтому третья запись не отвечает указанному условию.

Таким образом, указанным условиям отвечает только запись № 2.

Ответ: 2.

Задача 4. На олимпиаде по английскому языку предлагались задания трех типов; А, В и С. Итоги олимпиады были оформлены в таблицу, в которой было отражено, сколько заданий каждого типа выполнил каждый участник, например:

Фамилия, имя участника	А	В	С
Быкова Елена	3	1	1
Тихомиров Сергей	3	2	1

За правильное выполнение задания типа А участнику начислялся 1 балл, за выполнение задания типа В — 3 балла и за С — 5 баллов. Победитель определялся по сумме набранных баллов. При этом у всех участников сумма баллов оказалась разной. Какой запрос достаточно выполнить для определения победителя олимпиады:

1) отсортировать таблицу по убыванию значения столбца С и взять первую строку;

2) отсортировать таблицу по возрастанию значений выражения $A + B + C$ и взять первую строку;

3) отсортировать таблицу по убыванию значений выражения $A + 3B + 5C$ и взять первую строку;

4) отсортировать таблицу по возрастанию значений выражения $A + 3B + 5C$ и взять первую строку [4].

Решение.

Рассмотрим каждый из предлагаемых запросов:

1) первый и второй запросы неверно рассчитывают количество баллов (верная формула: $A + 3B + 5C$);

2) при использовании третьего запроса будет верно рассчитано количество баллов для каждого участника, а при сортировке по убыванию в первой строке будет получено число баллов участника, набравшего наибольшее количество баллов;

3) при использовании четвертого запроса будет верно рассчитано количество баллов для каждого участника, однако при сортировке по убыванию в первой строке будет находиться число баллов участника, набравшего наименьшее количество баллов.

Таким образом, для определения победителя необходимо использовать третий запрос.

Ответ: 3.

Задача 5. Результаты тестирования представлены в таблице:

Фамилия	Пол	Математика	Русский язык	Химия	Информатика	Биология
Аганян	ж	82	56	46	32	70
Воронин	м	43	62	45	74	23
Григорчук	м	54	74	68	75	83
Роднина	ж	71	63	56	82	79
Сергеенко	ж	33	25	74	38	46
Черепанова	ж	18	92	83	28	61

Сколько записей в ней удовлетворяют условию: Пол = 'ж' ИЛИ Химия > Биология [6]?

Решение.

Условию Пол = 'ж' удовлетворяют записи таблицы, значения поля «Фамилия» для которых равны: Аганян, Роднина, Сергеенко, Черепанова.

Условию Химия > Биология удовлетворяют записи таблицы, значения поля «Фамилия» для которых равны: Воронин, Сергеенко, Черепанова.

Таким образом, условию Пол = 'ж' ИЛИ Химия > Биология удовлетворяет 5 записей.

Ответ: 5.

Задача 6. Сколько записей в нижеследующем фрагменте турнирной таблицы удовлетворяют условию: Место >= 3 И (Н > 2 ИЛИ МП > 10) (символ >= означает «больше или равно»)?

Место	Команда	В	Н	П	О	МЗ	МП
1	Боец	7	4	2	14	9	6
2	Авангард	4	1	1	15	4	4
3	Опушка	5	4	3	11	5	5
4	Звезда	6	3	4	10	11	12
5	Химик	3	4	1	11	2	11
6	Пират	2	3	2	10	12	11

Решение.

Раскроем скобки в заданном условии:

Место >= 3 И (Н > 2 ИЛИ МП > 10) ⇒

⇒ (Место >= 3 И Н > 2) ИЛИ (Место >= 3 И МП > 10).

Первой части условия удовлетворяют команды, которые находятся на 3-м, 4-м, 5-м и 6-м местах. Второй части условия удовлетворяют команды, находящиеся на 4-м, 5-м и 6-м местах. Таким образом, условию **(Место ≥ 3 И Н > 2) ИЛИ (Место ≥ 3 И МП > 12)** удовлетворяют команды, находящиеся на 3-м, 4-м, 5-м и 6-м местах, т. е. всего 4 записи таблицы.

Ответ: 4.

Задача 7. Представлена таблица базы данных «Классы школы»:

Класс	Кол_учеников	Староста
9	27	Иванов
10	26	Петров
8	30	Чаадаев
11	18	Сидоров
7	24	Лукин

Куда переместятся сведения о Чаадаеве после сортировки таблицы в порядке возрастания по полю «Класс»?

Решение.

В поле «Класс» хранится целое число. Выполним сортировку данных в таблице по этому полю:

Класс	Кол_учеников	Староста
7	24	Лукин
8	30	Чаадаев
9	27	Иванов
10	26	Петров
11	18	Сидоров

Таким образом, сведения о Чаадаеве переместились на одну строку вверх.

Ответ: на одну строку вверх.

Задача 8. Таблица «Гараж», наряду с другими, имеет поля с названиями «Тип автомобиля» и «Мощность». В этой таблице находятся 25 записей о грузовых автомобилях и автобусах с мощностью двигателя в 100, 150 и 200 лошадиных сил. Количество записей N , удовлетворяющих различным запросам, приведено в следующей таблице:

Запрос	N
Мощность = 100 ИЛИ Тип автомобиля = грузовой	18
Неверно, что (Мощность = 200 И Тип автомобиля = грузовой)	31
Мощность = 150 И Тип автомобиля = автобус	4

Определите количество записей, удовлетворяющих запросу: «Мощность $\neq 200$ ».

Решение.

Пусть для обозначения количества автомобилей определенного типа используется формат: $\langle \text{Тип автомобиля} \rangle \langle \text{мощность} \rangle$, например: Г100 — количество грузовых автомобилей мощностью 100 л/с.

Используя запросы и значения N , составим уравнения:

1) в условии задачи указано, что в базе находится информация всего о 25 автомобилях, поэтому:

$$\Gamma 100 + \Gamma 150 + \Gamma 200 + A100 + A150 + A200 = 25; \quad (3.3)$$

2) количество записей, удовлетворяющих первому запросу — объединению результатов двух запросов: **Мощность = 100** и **Тип автомобиля = грузовой**:

$$\Gamma 100 + \Gamma 150 + \Gamma 200 + A100 = 18; \quad (3.4)$$

3) из второго запроса следует, что количество всех автомобилей, за исключением грузового с двигателем мощностью 200 л/с, равно 31:

$$\Gamma 100 + \Gamma 150 + A100 + A150 + A200 = 31; \quad (3.5)$$

4) из третьего запроса следует, что количество автобусов с двигателем мощностью 100 л/с равно 4:

$$A100 = 4.$$

Для решения задачи необходимо найти количество записей, удовлетворяющих запросу **Мощность $\neq 200$** , т. е. вычислить сумму $\Gamma 100 + \Gamma 150 + A100 + A150$.

Подставим (3.5) в (3.3) и получим, что $\Gamma 200 = 6$. Это значение подставим в (3.4) и (3.3) и запишем новые выражения:

$$\Gamma 100 + \Gamma 150 + A100 = 12; \quad (3.6)$$

$$\Gamma 100 + \Gamma 150 + A100 + A200 = 15. \quad (3.7)$$

Подставляя (3.6) в (3.7), получим, что $A200 = 3$.

Зная $A200$ и $\Gamma 200$, можно вычислить сумму $\Gamma 100 + \Gamma 150 + A100 + A150$:

$$\Gamma 100 + \Gamma 150 + A100 + A150 = 25 - A200 - \Gamma 200 = 25 - 3 - 4 = 18.$$

Ответ: 18.

3.3. Моделирование

Модель — это материальный или мысленно представляемый объект, который в процессе изучения замещает оригинал, сохраняя некоторые важные для данного исследования типичные свойства объекта. Под «объектом» при этом понимается любой материальный предмет, процесс или явление.

Обычно модели применяются, когда проведение экспериментов с реальным объектом сопряжено с определенными трудностями или невозможно по каким-то причинам (из-за большой продолжительности эксперимента во времени, возможности повреждения объекта, его опасности для исследователя и т. д.).

Моделирование — это процесс изучения строения и свойств оригинала с помощью модели. Процесс моделирования состоит из следующих этапов:

- 1) исследование свойств реального объекта;
- 2) построение модели, которая учитывает наиболее существенные для проводимого исследования свойства реального объекта;
- 3) изучение модели, проведение экспериментов с ней;
- 4) получение знаний о реальном объекте.

Существует два вида моделей: предметные и информационные.

Предметные модели воспроизводят геометрические, физические и другие свойства объекта в материальной форме (модель атома, солнечной системы и т. д.).

Информационные модели представляют объекты и процессы в образной или знаковой форме.

Виды информационных моделей:

- 1) текст (в том числе программа на некотором языке программирования);
- 2) математическая формула;
- 3) таблица;
- 4) блок-схема (например, блок-схема алгоритма);
- 5) граф.

Граф — это совокупность объектов со связями между ними.

Существуют ориентированные и неориентированные графы. *Неориентированный граф* G — это упорядоченная пара $G = (V, E)$, где V — множество *вершин*, или *узлов*, а E — множество неупорядоченных пар вершин, называемых *ребрами*.

Ориентированный граф (сокращенно — *орграф*) G — это упорядоченная пара $G = (V, A)$, где V — множество вершин, или узлов, а A — множество упорядоченных пар различных вершин, называемых

дугами, или *ориентированными ребрами*. *Дуга* — это упорядоченная пара вершин (v, w) , где вершину v называют *началом*, а w — *концом дуги* (дуга $v \rightarrow w$ ведет от вершины v к вершине w). Если v_1, v_2 — вершины, а $e = (v_1, v_2)$ — соединяющее их ребро, то вершина v_1 и ребро e *инцидентны* (связаны), а вершина v_2 и ребро e также инцидентны.

Способы представления графа:

1) *матрица смежности* — таблица, в которой столбцы и строки соответствуют вершинам графа. В каждой ячейке этой таблицы записывается число, определяющее наличие связи от вершины v_i к вершине v_j (или наоборот) и указывающее *вес* данной связи;

$$M_{i,j} = \begin{cases} 1, & \text{если вершина } v_i \text{ смежна с вершиной } v_j; \\ 0, & \text{в противном случае;} \end{cases}$$

2) *матрица инцидентности* — таблица, в которой каждая строка соответствует определенной вершине графа, а столбцы — связям между вершинами графа. Для неориентированного графа матрица инцидентности записывается следующим образом:

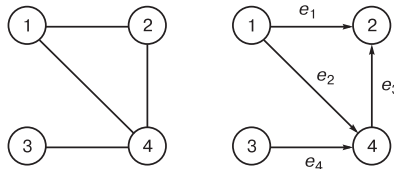
$$H_{i,j} = \begin{cases} 1, & \text{если вершина } v_i \text{ инцидентна ребру } e_j; \\ 0, & \text{в противном случае.} \end{cases}$$

Для ориентированного графа запись матрицы инцидентности имеет вид:

$$H_{i,j} = \begin{cases} 1, & \text{если вершина } v_i \text{ инцидентна ребру } e_j \text{ и является его концом;} \\ -1, & \text{если вершина } v_i \text{ инцидентна ребру } e_j \text{ и является его началом;} \\ 0, & \text{в противном случае.} \end{cases}$$

Граф можно задать также посредством списков пар вершин, соединенных ребрами (или дугами).

Задача 1. Постройте матрицы смежности, матрицы инцидентности и списки пар вершин для графов, приведенных на рисунках:



Решение.

Прежде всего сделаем важное замечание: для ориентированного графа стрелка от вершины v_1 к вершине v_2 указывает на наличие связи между первой и второй вершинами, но не обратной связи между второй и первой вершинами.

Матрица смежности
неориентированного графа:

$$M = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

Матрица смежности
ориентированного графа:

$$M = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

Матрица инцидентности
неориентированного графа:

$$H = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix}$$

Матрица инцидентности
ориентированного графа:

$$H = \begin{pmatrix} -1 & -1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 1 & -1 & 1 \end{pmatrix}$$

Список пар вершин неориентированного графа: $\{1, 2\}, \{1, 4\}, \{2, 4\}, \{3, 4\}$.*

Список пар вершин ориентированного графа: $((1, 2), (1, 4), (4, 2), (3, 4))$.

Задача 2. Путешественник пришел в 08:00 на автостанцию населенного пункта КАЛИНИНО и обнаружил следующее расписание автобусов:

Пункт отправления	Пункт прибытия	Время отправления	Время прибытия
КАМЫШИ	КАЛИНИНО	08:15	09:10
КАЛИНИНО	БУКОВОЕ	09:10	10:15
РАКИТИНО	КАМЫШИ	10:00	11:10
РАКИТИНО	КАЛИНИНО	10:05	12:25
РАКИТИНО	БУКОВОЕ	10:10	11:15
КАЛИНИНО	РАКИТИНО	10:15	12:35
КАЛИНИНО	КАМЫШИ	10:20	11:15
БУКОВОЕ	КАЛИНИНО	10:35	11:40
КАМЫШИ	РАКИТИНО	11:25	12:30
БУКОВОЕ	РАКИТИНО	11:40	12:40

* В фигурных скобках записываются пары вершин для неориентированного графа.

Определите самое раннее время, когда путешественник сможет оказаться в пункте РАКИТИНО согласно этому расписанию [7].

Решение.

По приведенному расписанию автобусов построим схему, которая отражает возможные связи между населенными пунктами (рис. 3.5).

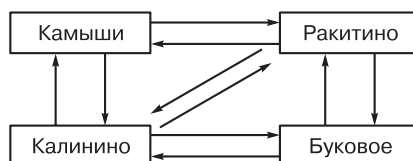


Рис. 3.5. Схема маршрутов между населенными пунктами

Из пункта Калинино в пункт Ракитино ведет несколько маршрутов:

1) прямой маршрут Калинино → Ракитино: время отправления из Калинино — 10:15, время прибытия в Ракитино — 12:35;

2) маршрут Калинино → Камыши → Ракитино: Калинино → Камыши (10:20 – 11:15), Камыши → Ракитино (11:25 – 12:30), время прибытия в Ракитино — 12:30;

3) маршрут Калинино → Буковое → Ракитино: Калинино → Буковое (09:10 – 10:15), Буковое → Ракитино (11:40 – 12:40), время прибытия в Ракитино — 12:40.

Следовательно, самое раннее время, в которое путешественник сможет оказаться в Ракитино, — 12:30.

Ответ: 12:30.

Задача 3. Между четырьмя крупными аэропортами, обозначенными кодами DLU, IGT, OPK и QLO, ежедневно выполняются авиарейсы. Приведен фрагмент расписания перелетов между этими аэропортами:

Аэропорт вылета	Аэропорт прилета	Время вылета	Время прилета
QLO	IGT	06:20	08:35
IGT	DLU	10:25	12:35
DLU	IGT	11:45	13:30
OPK	QLO	12:15	14:25
QLO	DLU	12:45	16:35
IGT	QLO	13:15	15:40
DLU	QLO	13:40	17:25
DLU	OPK	15:30	17:15
QLO	OPK	17:35	19:30
OPK	DLU	19:40	21:55

Путешественник находится в аэропорту DLU в полночь (0:00). Определите самое раннее время, когда он сможет оказаться в аэропорту QLO [6].

Решение.

Построим по указанному расписанию схему маршрутов между аэропортами (рис. 3.6).

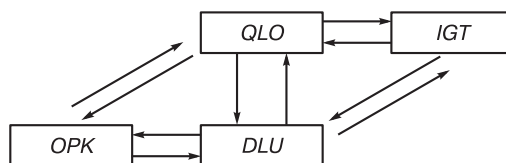


Рис. 3.6. Схема маршрутов между аэропортами

Из аэропорта DLU в аэропорт QLO можно попасть, используя несколько маршрутов:

- 1) DLU → QLO: 13:40 – 17:25, время прибытия 17:25;
- 2) DLU → ОРК → QLO: 15:30 – 17:15, 12:15 – 14:25. Однако данный маршрут не может быть использован, так как время прибытия в аэропорт ОРК с рейса DLU → ОРК более позднее, чем время отправления рейса ОРК → QLO;
- 3) DLU → IGT → QLO: 11:45 – 13:30, 13:15 – 15:40. Однако данный маршрут также не может быть использован, так как время прибытия в аэропорт IGT с рейса DLU → IGT более позднее, чем время отправления рейса IGT → QLO.

Таким образом, путешественник сможет оказаться в аэропорту QLO только в 17:25.

Ответ: 17:25.

Задача 4. Грунтовая дорога проходит последовательно через населенные пункты A , B , C и D . При этом длина дороги между A и B равна 80 км, между B и C — 50 км, а между C и D — 10 км.

Между A и C построили новое асфальтовое шоссе длиной 40 км. Оцените минимально возможное время движения велосипедиста из пункта A в пункт B , если его скорость по грунтовой дороге — 20 км/ч, а по шоссе — 40 км/ч [5].

Решение.

Из пункта A в пункт B существует два маршрута (см. рис. 3.7):

- 1) $A \rightarrow B$;
- 2) $A \rightarrow C \rightarrow B$.

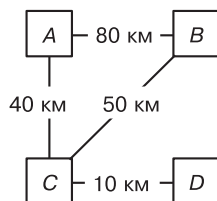


Рис. 3.7. Схема маршрутов между населенными пунктами

Рассчитаем время движения по обоим этим маршрутам:

1) $t = 80 \text{ (км)} / 20 \text{ (км/ч)} = 4 \text{ часа}$;

2) $t = 40 \text{ (км)} / 40 \text{ (км/ч)} + 50 \text{ (км)} / 20 \text{ (км/ч)} = 3,5 \text{ часа}$.

Таким образом, минимально возможное время движения — 3,5 часа.

Ответ: 3,5 часа.

Задача 5. Таблица стоимости перевозок устроена следующим образом: числа, стоящие на пересечениях строк и столбцов таблиц, обозначают стоимость проезда между соответствующими соседними станциями. Если пересечение строки и столбца пусто, то станции не являются соседними.

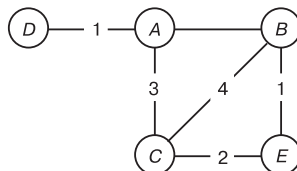
	A	B	C	D	E
A			3	1	
B			4		1
C	3	4			2
D	1				
E		1	2		

Определите, выполняется ли для приведенной таблицы условие: «Минимальная стоимость проезда из A в B не больше 6».

Стоимость проезда по маршруту складывается из стоимостей проезда между соответствующими соседними станциями [4].

Решение.

Приведенная таблица стоимости проезда представляет собой матрицу смежности графа. Построим граф по данной матрице:



Рассмотрим этот граф и возможные маршруты из вершины A в вершину B :

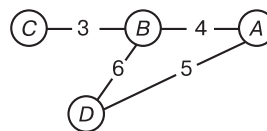
- 1) $A \rightarrow C \rightarrow B = 3 + 4 = 7$, стоимость проезда = 7;
- 2) $A \rightarrow C \rightarrow E \rightarrow B = 3 + 2 + 1 = 6$, стоимость проезда = 6.

Для приведенной таблицы указанному условию удовлетворяет маршрут $A \rightarrow C \rightarrow E \rightarrow B$.

Ответ: для приведенной таблицы условие выполняется.

Задача 6. В таблице приведена стоимость перевозок между соседними железнодорожными станциями. Определите, соответствует ли приведенная схема этой таблице [2].

	A	B	C	D
A		4		5
B	4		3	6
C		3		
D	5	6		



Решение.

Исходя из таблицы, можно сделать следующие выводы:

- 1) вершины A и B должны быть соединены, причем вес связи равен 4;
 - 2) должны быть соединены вершины A и D , вес связи — 5;
 - 3) должны быть соединены вершины B и C , вес связи — 3;
 - 4) должны быть соединены вершины B и D , вес связи — 6;
- Таким образом, приведенная таблица соответствует схеме.

Ответ: приведенная таблица соответствует схеме.

Задача оптимизации

Задача моделирования объекта тесно связана с задачей оптимизации. Под оптимизацией в данном случае понимается нахождение экстремума (минимума или максимума) функции n переменных $f(X) = f(x_1, \dots, x_n)$.

Для решения задачи оптимизации необходимо вначале построить математическую модель исследуемого объекта, включающую:

- 1) управляемые переменные, значения которых подлежат оптимизации;
- 2) ограничения на область изменения управляемых переменных;
- 3) числовой критерий оптимизации — описание целевой функции.

Практических применений задачи оптимизации существует множество: это различные экономические приложения, связанные с ре-

шением задач планирования, задачи логистики, связанные с оптимизации цепей поставок, и т. д.

Теорию и методы решения задач нахождения экстремумов функций изучает *математическое программирование*. Если же ограничения и целевая функция заданы в виде линейных уравнений и неравенств, то для решения такой задачи оптимизации применяются методы *линейного программирования*.

Задачу линейного программирования можно сформулировать следующим образом: требуется найти максимум функции $F = a_{0,1}x_1 + a_{0,2}x_2 + \dots + a_{0,n}x_n + b_0$ при условии:

$$\begin{cases} a_{1,1}x_1 + a_{1,2}x_2 + \dots + a_{1,n}x_n \leq b_1; \\ a_{2,1}x_1 + a_{2,2}x_2 + \dots + a_{2,n}x_n \leq b_2; \\ \dots \\ a_{m,1}x_1 + a_{m,2}x_2 + \dots + a_{m,n}x_n \leq b_m; \\ x_1 \geq 0, x_2 \geq 0, \dots, x_n \geq 0. \end{cases}$$

Указанные ограничения называются *условиями неотрицательности*.

Графический метод решения задач линейного программирования

Графический метод применяется, когда количество переменных равно двум (реже — трем), при этом количество ограничений может быть любым. В этом случае целевая функция записывается следующим образом:

$$F(x_1, x_2) = c_1 \cdot x_1 + c_2 \cdot x_2.$$

Этапы решения задачи:

1. На координатной плоскости отобразите область допустимых решений, представляющую собой пересечение полуплоскостей, определяемых неравенствами системы ограничений. Решению задачи линейного программирования всегда соответствует одна или несколько точек области решений.

2. Постройте в плоскости вектор-градиент, начало которого — точка с координатами $(0, 0)$, а конец — точка с координатами (c_1, c_2) , где c_1 и c_2 — коэффициенты при переменных в целевой функции.

3. Приравняйте целевую функцию некоторой константе (например, 0):

$$f(x_1, x_2) = c_1 \cdot x_1 + c_2 \cdot x_2 = 0.$$

Данное уравнение является уравнением прямой уровня. Постройте ее на координатной плоскости. Перемещайте прямую уровня по направлению вектора-градиента до тех пор, пока прямая полностью

не выйдет за границу области допустимых значений. В результате будет найдена точка (или несколько точек), в которой целевая функция принимает максимальное значение. Если же требуется найти точку, в которой $f(x_1, x_2)$ принимает минимальное значение, то прямую уровня нужно перемещать в направлении, противоположном вектору-градиенту.

4. Определите прямые, на пересечении которых находится найденная граничная точка. Составьте из уравнений прямых систему уравнений и решите ее. Полученные значения x_1 и x_2 определяют искомые координаты точки, в которой целевая функция принимает максимум (минимум).

При решении задачи линейного программирования могут быть следующие особенности:

1) если прямая уровня при перемещении не вышла за область допустимых решений (т. е. область допустимых решений не ограничена), то максимум (минимум) целевой функции не существует (случай *a* на рисунке);

2) если прямая уровня параллельна одной из границ области допустимых решений, то оптимальное значение целевой функции достигается в любой точке этой границы, а количество оптимальных значений соответствует бесконечному множеству (линия *A* — линия уровня — случай *б* на рисунке);

3) если полуплоскости, задаваемые ограничениями, не имеют общих точек, то говорят, что ограничения несовместны; тогда множество значений пусто, а задача не имеет решения (случай *в* на рис. 3.8).

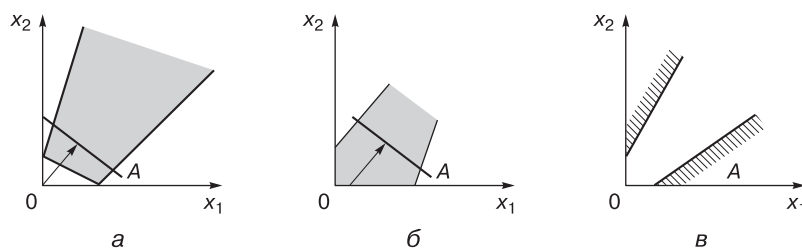


Рис. 3.8. Особые случаи при решении задачи оптимизации

Рассмотрим способ определения полуплоскости, которую описывает заданное неравенство.

1. Получите уравнение прямой, заменив знак неравенства на знак равенства, и постройте прямую на координатной плоскости.

2. Выберите любую точку $A(x_1, x_2)$ координатной плоскости и подставьте ее координаты в неравенство.

3. Если неравенство выполняется, то точка $A(x_1, x_2)$ является допустимым решением, а полуплоскость, содержащая точку, удовлетворяет неравенству.

Допустим, что одним из неравенств системы ограничений является следующее: $4x_1 - 2x_2 \geq 1$. Запишите уравнение прямой в отрезках ($x_1/2 + x_2/4 = 1$) и постройте эту прямую. Проверьте точку с координатами $(0, 0)$. Подставьте ее в неравенство: $4 \cdot 0 - 2 \cdot 0 = 0 \geq 1$. Неравенство не выполняется, следовательно, неравенству $4x_1 - 2x_2 \geq 1$ соответствует полуплоскость, не содержащая точку $(0, 0)$. Полуплоскость, соответствующая данному неравенству, показана на рис. 3.9

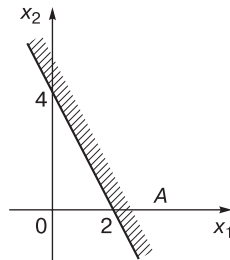


Рис. 3.9. Решение задачи оптимизации

Задача 7. Определите максимальное значение целевой функции $f(x_1, x_2) = x_1 + 2x_2 \rightarrow \max$ при системе ограничений:

$$\begin{cases} 3x_1 - 2x_2 \leq 6; \\ -x_1 + 2x_2 \leq 4; \\ 3x_1 + 2x_2 \leq 12; \\ x_1 \geq 0, x_2 \geq 0. \end{cases}$$

Решение.

Определим область допустимых значений. По условию задачи, $x_1 \geq 0, x_2 \geq 0$, тогда в качестве возможных решений будут рассматриваться только точки, которые принадлежат первой четверти координатной плоскости.

I. Рассмотрим первое неравенство: $3x_1 - 2x_2 \leq 6$. Разделим каждый член этого неравенства на 6: $x_1/2 + x_2/(-3) \leq 1$. Заменяя знак неравенства знаком равенства, получим уравнение прямой в отрезках. Прямая, соответствующая уравнению $x_1/2 + x_2/(-3) = 1$, пересекается с осью абсцисс в точке $(2, 0)$, а с осью ординат — в точке $(0, -3)$. Построим эту прямую. Наше неравенство определяет точки, лежащие выше построенной прямой.

II. Рассмотрим теперь неравенство $-x_1 + 2x_2 \leq 4$. Разделим каждый его член на 4: $x_1/(-4) + x_2/2 \leq 1$. Заменяя знак неравенства

знаком равенства, получим уравнение прямой в отрезках и построим эту прямую. В результате получим, что рассматриваемое неравенство определяет точки, лежащие ниже построенной прямой.

III. Рассмотрим неравенство $3x_1 + 2x_2 \leq 12$. Разделим каждый его член на 12: $x_1 / 4 + x_2 / 6 \leq 1$. Заменяя знак неравенства знаком равенства, получим уравнение прямой в отрезках и построим эту прямую. Получим, что анализируемое неравенство определяет точки, лежащие ниже построенной прямой.

IV. Рассмотрим исходную функцию. Приравняем ее к нулю: $x_1 + 2x_2 = 0$. В результате получим уравнение прямой уровня на плоскости.

V. Построим на плоскости вектор-градиент. Его начало находится в точке $(0, 0)$, а конец — в точке $(1, 2)$.

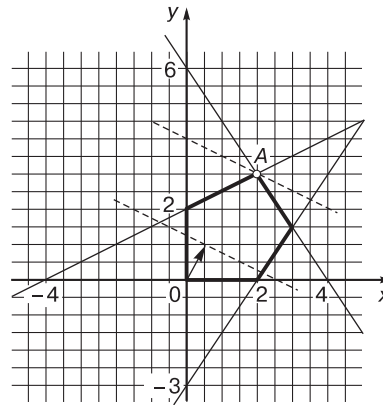
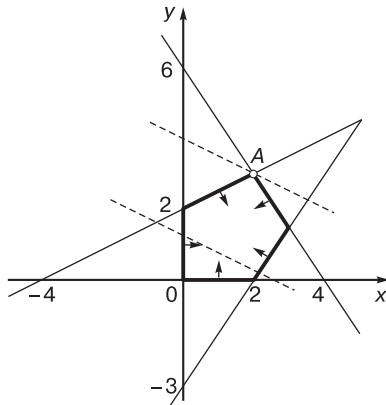
Поскольку необходимо определить максимальное значение функции, будем перемещать прямую до последнего касания обозначенной области (на графике она обозначена пунктирной линией).

Прямая уровня пересекает область в точке A , которая является точкой пересечения прямых 2 и 3. Тогда координаты данной точки можно вычислить из уравнений этих прямых:

$$\begin{cases} -x_1 + 2x_2 = 4, \\ 3x_1 + 2x_2 = 12. \end{cases}$$

Решив эту систему уравнений, получим, что $x_1 = 2$, $x_2 = 3$.

Определим теперь максимальное значение целевой функции: $f(x_1, x_2) = 1 \cdot 2 + 2 \cdot 3 = 8$.



Ответ: $\max(f(x_1, x_2)) = 8$.

Табличный симплекс-метод

Одним из методов решения задачи линейного программирования является табличный симплекс-метод. Основная идея этого метода состоит в переходе от одного допустимого базисного решения к другому так, чтобы значения целевой функции при этом непрерывно возрастали (для задачи поиска максимума). Табличный метод также называют методом последовательного улучшения оценки.

Рассмотрим этапы решения задачи поиска максимума (минимума) функции на основе табличного симплекс-метода.

1. Если в исходной задаче необходимо найти минимум, то знаки коэффициентов целевой функции f меняются на противоположные, в результате чего задача поиска минимума сводится к поиску максимума, а после определения минимального значения функции нужно умножить его на -1 . При этом если в списке ограничений содержится условие со знаком \geq , то его нужно привести к условию со знаком \leq путем умножения на -1 . В результате все коэффициенты условия изменятся на противоположные.

2. Составление симплекс-таблицы:

2.1) в столбец «Б» записываются переменные, принимаемые за *базисные*. Базисной называется переменная, если она входит только в одно уравнение системы и нет такого уравнения, в которое не входила бы хотя бы одна из базисных переменных;

2.2) столбец «С» — это столбец *свободных членов*. В него записываются правые части ограничений. Далее идут столбцы коэффициентов x_i при i -й переменной, находящейся в каждом уравнении;

2.3) в последнюю строку симплекс-таблицы заносятся коэффициенты целевой функции с противоположными знаками.

Б	С	x_1	...	x_n	x_{n+1}	...	x_{n+m}
x_{n+1}	b_1	$a_{1,1}$...	$a_{1,n}$	$a_{1,n+1}$...	$a_{1,n+m}$
x_{n+2}	b_2	$a_{2,1}$...	$a_{2,n}$	$a_{2,n+1}$...	$a_{2,n+m}$
...
x_{n+m}	b_m	$a_{r,1}$...	$a_{r,n}$	$a_{n+m,n+1}$...	$a_{n+m,n+m}$
f	f_0	c_1	...	c_n	0	...	0

3. Проверка решения на допустимость. Просматриваются элементы столбца «С» (свободные члены). Если среди них нет отрицательных, то допустимое решение найдено и выполняется переход на следующий шаг. Иначе:

3.1) из отрицательных элементов столбца «С» выбирается элемент, максимальный по модулю, — он задает ведущую строку k ;

3.2) в ведущей строке определяется максимальный по модулю отрицательный элемент $a_{k,l}$; он определяет ведущий столбец l и является ведущим элементом;

3.3) переменная, соответствующая ведущей строке, исключается из базиса, а переменная, соответствующая ведущему столбцу, включается в базис;

3.4) выполняется пересчет симплекс-таблицы:

- разделите ведущую строку на значение ведущего элемента — ведущий коэффициент станет равным единице;
- умножьте ведущую строку на такой коэффициент, чтобы после ее сложения с другими строками в ведущем столбце появились нули (указанным образом обрабатывается в том числе и строка, относящаяся к коэффициентам целевой функции);

3.5) если после пересчета таблицы в столбце свободных членов остались отрицательные элементы, то выполняются действия, описанные на данном шаге. В противном случае выполняется переход на следующий шаг.

Примечание. Если среди свободных членов есть отрицательные элементы, а в соответствующей строке таких элементов нет, то задача решений не имеет, так как условия задачи несовместны.

4. Проверка найденного решения на оптимальность: просматриваются элементы таблицы, находящиеся в строке f (элемент f_0 не учитывается). Если в строке f есть отрицательные элементы, то нужно улучшить решение:

4.1) среди отрицательных элементов строки f (исключая f_0) выбирается максимальный по модулю. Столбец l , в котором находится элемент, называется ведущим. Для определения ведущей строки вычисляется отношение соответствующего свободного члена и элемента из ведущего столбца, при условии что они неотрицательны:

$$b_k / a_{k,l} = \min \{ b_i / a_{i,l} \} \text{ при } a_{i,l} > 0, b_i > 0.$$

Строка k , для которой это отношение минимально, — ведущая. Элемент $a_{k,l}$ — ведущий (разрешающий);

4.2) переменная, соответствующая ведущей строке (x_k), исключается из базиса, а переменная, соответствующая ведущему столбцу (x_l), включается в базис;

4.3) выполняется пересчет симплекс-таблицы (метод пересчета описан в третьем шаге);

4.4) если в новой таблице после ее перерасчета в строке f остались отрицательные элементы, то выполняется переход на начало шага 4.

5. Если невозможно найти ведущую строку, так как в ведущем столбце нет положительных элементов, то функция в области допустимых решений задачи не ограничена — алгоритм завершает работу.

Если в строке f и в столбце свободных членов все элементы положительные, то найдено оптимальное решение.

Задача 8. Решить задачу линейного программирования табличным симплекс-методом:

$$\begin{aligned} f(x_1, x_2) &= 7x_1 + 5x_2 \rightarrow \max; \\ \begin{cases} x_1 + x_2 \leq 3; \\ x_1 + 5x_2 \leq 5; \\ 2x_1 + x_2 \leq 4. \end{cases} \end{aligned}$$

Решение.

Приведем заданную систему неравенств к системе уравнений путем введения дополнительных переменных: добавим в каждое уравнение переменную с коэффициентом, равным единице, которая будет отсутствовать в остальных уравнениях.

$$\begin{cases} 1x_1 + 1x_2 + 1x_3 + 0x_4 + 0x_5 = 3; \\ 1x_1 + 5x_2 + 0x_3 + 1x_4 + 0x_5 = 5; \\ 2x_1 + 1x_2 + 0x_3 + 0x_4 + 1x_5 = 4. \end{cases}$$

Заполним *симплекс-таблицу*. В строку f внесем коэффициенты целевой функции с противоположными знаками.

			1	2	3	4	5
	Б	С	x_1	x_2	x_3	x_4	x_5
1	x_3	3	1	1	1	0	0
2	x_4	5	1	5	0	1	0
3	x_5	4	2	1	0	0	1
4	f	0	-7	-5	0	0	0

Все свободные члены в этой таблице положительны, следовательно, найдено допустимое решение.

Выполним преобразование симплекс-таблицы.

1. Проверим критерий оптимальности (рассматриваем все коэффициенты строки целевой функции f): признак оптимальности таблицы — неотрицательность коэффициентов при небазисных переменных. В нашем случае этот критерий не выполнен, поэтому выполняем переход на следующий шаг.

2. Среди отрицательных коэффициентов выбираем коэффициент, максимальный по модулю (второй столбец, элемент -7). Столбец, в котором находится данный коэффициент, определяем как ведущий, причем в нем есть положительные элементы. Определим теперь ведущую строку. Вычислим отношение каждого элемента ведущего столбца к соответствующему столбцу свободных членов и определим минимальное отношение: $\min \{3/5, 5/1, 4/2\} = 4/2 = 2$. Получим, что ведущей будет третья строка.

3. Выполним преобразование таблицы: разделим третью строку таблицы на 2 и вычтем результат из первой и второй строк. Умножим третью строку на 7 и сложим ее с четвертой строкой (f). В вычислениях участвуют также и элементы столбца «С».

В результате новыми базисными переменными становятся x_3 , x_4 и x_1 (x_1 добавляется, так как минимальным выбран элемент первого столбца, соответствующего вектору x_1).

			1	2	3	4	5
	Б	С	x_1	x_2	x_3	x_4	x_5
1	x_3	1	0	0,5	1	0	-0,5
2	x_4	3	0	4,5	0	1	-0,5
3	x_1	2	1	0,5	0	0	0,5
4	f	14	0	-1,5	0	0	3,5

4. Вновь проверим критерий оптимальности. Отрицательный коэффициент только один — в столбце x_2 .

5. Вычислим: $\min \{1/0,5, 3/4,5, 2/0,5\} = 3/4,5 = 2/3$. Разрешающим элементом будет второй элемент второго столбца — $2/3$, а новыми базисными переменными становятся x_3 , x_2 и x_1 .

6. Выполним преобразование таблицы: разделим вторую строку на $4,5$, после чего умножим ее на $-0,5$ и сложим с первой и второй строками. Умножим вторую строку на $1,5$ и сложим ее с четвертой строкой (f).

			1	2	3	4	5
	Б	С	x_1	x_2	x_3	x_4	x_5
1	x_3	$2/3$	0	0	1	$-1/9$	$-4/9$
2	x_2	$2/3$	0	1	0	$2/9$	$-1/9$
3	x_1	$5/3$	1	0	0	$-1/9$	$5/9$
4	f	15	0	0	0	$1/3$	$10/3$

На этот раз отрицательных оценок нет, т. е. критерий оптимальности выполнен. Оптимальная оценка: $\max(f(x_1, x_2)) = 15$ при $x_1 = 5/3$, $x_2 = 2/3$.

Ответ: $\max(f(x_1, x_2)) = 15$.

Задача 9. Решить задачу линейного программирования табличным симплекс-методом:

$$\begin{aligned} f(x_1, x_2) &= -3x_1 + 6x_2 \rightarrow \max \\ \begin{cases} 2x_1 + 3x_2 \leq 7; \\ x_1 + 5x_2 \geq 8. \end{cases} \end{aligned}$$

Решение.

1. Приведем систему ограничений к каноническому виду, преобразуя неравенства в равенства с добавлением дополнительных переменных. Во втором неравенстве стоит знак \geq , поэтому при переходе к равенству знаки всех его коэффициентов и свободных членов меняются на противоположные. В результате система ограничений получит вид:

$$\begin{cases} 2x_1 + 3x_2 + 1x_3 + 0x_4 = 7; \\ -1x_1 + (-5)x_2 + 0x_3 + x_4 = -8. \end{cases}$$

Заполним симплекс-таблицу. В строку f при этом заносим коэффициенты целевой функции, взятые с противоположным знаком.

			1	2	3	4
	Б	С	x_1	x_2	x_3	x_4
1	x_3	7	2	3	1	0
2	x_4	-8	-1	-5	0	1
3	f	0	3	-6	0	0

2. Таблица в столбце свободных членов содержит отрицательные элементы. Выполним ее преобразование. Определим среди отрицательных элементов максимальный по модулю: -8, следовательно, ведущей будет вторая строка. Во второй строке максимальный по модулю отрицательный элемент — -5, он находится во втором столбце, который и будет ведущим. Переменная в ведущей строке исключается из базиса, а переменная, соответствующая ведущему столбцу, включается в базис. Учитывая это, пересчитываем симплекс-таблицу.

Разделим ведущую строку на -5. Умножим вторую строку на 6 и сложим ее со строкой f . Умножим вторую строку на -3 и сложим ее с первой строкой.

			1	2	3	4
	Б	С	x_1	x_2	x_3	x_4
1	x_3	2,2	1,4	0	1	0,6
2	x_2	1,6	0,2	1	0	-0,2
3	f	9,6	4,2	0	0	-1,2

3. В столбце свободных членов нет отрицательных элементов, т. е. найдено допустимое решение. Однако в строке f имеются отрицательные элементы, т. е. полученное решение можно улучшить. Определим ведущий столбец. Максимальный по модулю отрицательный элемент в строке $f = -1,2$, значит, ведущий столбец — четвертый.

Вычислим $\min \{2,2/0,6\} = 11/3$. Следовательно, ведущей строкой будет первая, а ведущий элемент — 0,6. Переменная в ведущей строке исключается из базиса, а переменная, соответствующая ведущему столбцу, включается в базис. Учитывая это, пересчитываем симплекс-таблицу.

Разделим ведущую строку на 0,6. Умножим первую строку на $-1,2$ и сложим ее со строкой f . Умножим первую строку на $-0,2$ и сложим ее со второй строкой.

			1	2	3	4
	Б	С	x_1	x_2	x_3	x_4
1	x_4	3,667	2,333	0	1,667	1
2	x_2	2,333	0,667	1	0	0
3	f	14	7	0	2	0

В строке f нет отрицательных элементов, следовательно, найдено оптимальное решение: $\max(f(x_1, x_2)) = 14$, при $x_1 = 0$, $x_2 = 2,3$.

Ответ: $\max(f(x_1, x_2)) = 14$.

Задача 10. Решить задачу линейного программирования табличным симплекс-методом:

$$f(x_1, x_2) = 2x_1 - 3x_2 \rightarrow \min$$

$$\begin{cases} 1x_1 + 5x_2 \leq 10; \\ 3x_1 + 2x_2 \leq 12; \\ 2x_1 + 4x_2 \leq 10. \end{cases}$$

Решение.

1. Приведем систему ограничений к каноническому виду, преобразуя неравенства в равенства с добавлением дополнительных переменных. В результате система ограничений примет вид:

$$\begin{cases} 1x_1 + 5x_2 + 1x_3 + 0x_4 + 0x_5 \leq 10; \\ 3x_1 + 2x_2 + 0x_3 + 1x_4 + 0x_5 \leq 12; \\ 2x_1 + 4x_2 + 0x_3 + 0x_4 + 1x_5 \leq 10. \end{cases}$$

В данной задаче необходимо найти *минимум* целевой функции, поэтому изменим знаки коэффициентов целевой функции F на противоположные:

$$f(x_1, x_2) = -2x_1 + 3x_2.$$

2. Заполним симплекс-таблицу; при этом в строку f заносятся коэффициенты целевой функции с противоположным знаком.

			1	2	3	4	5
	Б	С	x_1	x_2	x_3	x_4	x_5
1	x_3	10	1	5	1	0	0
2	x_4	12	3	2	0	1	0
3	x_5	10	2	4	0	0	1
4	f	0	2	-3	0	0	0

В столбце свободных членов отрицательных элементов нет, следовательно, найдено допустимое решение. Однако в строке f есть отрицательные элементы, поэтому полученное решение можно улучшить.

3. Определим ведущий столбец. Максимальный по модулю отрицательный элемент в строке $f = -3$, следовательно, ведущий столбец — второй. Ведущей же строкой будет та, для которой отношение свободного члена к соответствующему элементу ведущего столбца минимально. Тогда получаем, что ведущей строкой является первая, а ведущий элемент — 5.

4. Выполним преобразование симплекс-таблицы. Разделим ведущую строку на 5. Умножим первую строку на 3 и сложим ее со строкой f . Умножим первую строку на -2 и сложим ее со второй строкой. Умножим первую строку на -4 и сложим ее с третьей строкой.

			1	2	3	4	5
	Б	С	x_1	x_2	x_3	x_4	x_5
1	x_2	2	0,2	1	0,2	0	0
2	x_4	8	2,6	0	-0,4	1	0
3	x_5	2	1,2	0	-0,8	0	1
4	f	6	2,6	0	0,6	0	0

5. В строке f нет отрицательных элементов, поэтому оптимальное решение найдено. Исходная задача заключалась в поиске минимума целевой функции, поэтому для получения ответа необходимо инвертировать свободный член строки f : $f = -6$ при $x_1 = 0$, $x_2 = 2$.

Ответ: $\min(f(x_1, x_2)) = -6$.

Задача коммивояжера

Коммивояжер (путешествующий продавец) должен посетить N городов. Дана матрица C стоимости переходов между городами, число строк и столбцов которой соответственно равно N . На пересечении i -й строки и j -го столбца этой матрицы записана стоимость перехода

из города i в город j . В качестве такой стоимости может выступать расстояние между городами, стоимость перевозки товара или более сложный показатель. Задача оптимизации в данном случае состоит в поиске самого выгодного маршрута, проходящего между всеми городами, при этом:

- 1) в каждом городе коммивояжер должен побывать только один раз;
- 2) начальный и конечный города совпадают.

Для решения этой задачи применяется метод «ветвей и границ», алгоритм которого следующий.

1. Операция редукции матрицы стоимости:

1.1) определим в каждой строке матрицы стоимости C минимальный элемент, после чего вычтем его из каждого элемента строки;

1.2) определим в каждом столбце матрицы минимальный элемент, после чего вычтем его из каждого элемента столбца;

1.3) в результате будет получена матрица стоимости, каждая строка и каждый столбец которой содержат хотя бы один нулевой элемент. Подсчитаем сумму H констант приведения, т. е. сумму всех вычтенных элементов в строках и столбцах.

2. Вычислим для каждого нулевого элемента матрицы C коэффициент $d_{i,j}$, который равен сумме наименьшего элемента i -й строки (исключая элемент $C_{i,j} = 0$) и наименьшего элемента j -го столбца. Выберем максимальный из всех коэффициентов $d_{i,j}$. В маршрут коммивояжера (в *гамильтонов цикл*) тогда вносится дуга (i, j) .

3. Удалим из матрицы стоимости i -ю строку и j -й столбец и заменим на бесконечность значение элемента $C_{j,i}$ (дуга (i, j) включена в контур, поэтому обратный путь из j в i недопустим).

Далее шаги 1–3 повторяются, пока порядок матрицы не станет равным двум.

Наконец, в текущий ориентированный граф, представляющий собой часть маршрута коммивояжера, внесем две недостающие дуги, которые определяются матрицей порядка 2. В результате будет получен *гамильтонов цикл*. Итоговое значение H (суммы констант приведения) должно при этом совпасть с длиной результирующего гамильтонова цикла.

Примечания:

1) порядок квадратной матрицы — это количество ее строк (столбцов), причем матрица называется квадратной, если количество строк матрицы равно количеству столбцов;

2) гамильтонов цикл — это гамильтонов путь, начальная и конечная вершины которого совпадают; гамильтонов путь — это путь, содержащий каждую вершину графа ровно один раз.

Задача 11. Коммивояжер должен побывать в 5 городах. Для сокращения расходов он должен выбрать такой маршрут, чтобы посетить все города только один раз и вернуться в исходный с минимумом затрат. Исходный город — 1. Затраты на перемещение между городами заданы матрицей:

	1	2	3	4	5
1	M	19	35	8	22
2	2	M	11	1	25
3	14	7	M	29	5
4	15	10	19	M	12
5	16	5	5	8	M

Решите задачу методом ветвей и границ.

Решение.

Выполним редукцию матрицы по строкам. Для этого необходимо в каждой строке матрицы найти минимальный элемент и вычесть его из элементов рассматриваемой строки. Тогда в каждой строке полученной матрицы получится как минимум один ноль.

	1	2	3	4	5
1	M	11	27	0	14
2	1	M	10	0	24
3	9	2	M	24	0
4	5	0	9	M	2
5	11	0	0	3	M

Применим операцию редукции к столбцам. Найдем минимальный элемент в каждом столбце и вычтем из каждого элемента столбца найденный минимум. Получим матрицу:

	1	2	3	4	5
1	M	11	27	0	14
2	0	M	10	0	24
3	8	2	M	24	0
4	4	0	9	M	2
5	10	0	0	3	M

Сумма констант приведения определяет нижнюю границу H :

$$H = 8 + 1 + 5 + 10 + 5 + 1 = 30.$$

Для каждого нулевого элемента матрицы C определим сумму наименьшего элемента i -й строки и j -го столбца, в котором находится элемент:

	1	2	3	4	5
1	M	11	27	0	14
2	0	M	10	0	24
3	8	2	M	24	0
4	4	0	9	M	2
5	10	0	0	3	M

1) $d(1,4) = 11 + 0 = 11$;

2) $d(2,1) = 0 + 4 = 4$;

3) $d(2,4) = 0 + 0 = 0$;

4) $d(3,5) = 2 + 2 = 4$;

5) $d(4,2) = 2 + 0 = 2$;

6) $d(5,2) = 0 + 0 = 0$;

7) $d(5,3) = 0 + 9 = 9$.

Наибольшая сумма констант приведения равна $(11 + 0) = 11$ для ребра $(1,4)$. Включим ребро $(1,4)$ в маршрут и удалим из матрицы C первую строку и четвертый столбец. Заменяем в матрице элемент C_{41} на M , для исключения образования негамильтонова цикла (цикла $1 \rightarrow 4 \rightarrow 1$). Получим сокращенную матрицу (4×4) , которая подлежит операции приведения.

Сумма констант приведения сокращенной матрицы равна 0 ($H = 30$), т. е. изменения в матрицу не были внесены. В результате сокращенная матрица будет иметь вид:

	1	2	3	5
2	0	M	10	24
3	8	2	M	0
4	M	0	9	2
5	10	0	0	M

Заменяем все клетки полученной матрицы с нулевыми элементами на « M » (обозначение бесконечности) и определим для них сумму наименьшего элемента строки и столбца, в которой находится нулевой элемент:

1) $d(2,1) = 10 + 8 = 18$;

2) $d(3,5) = 2 + 2 = 4$;

3) $d(4,2) = 2 + 0 = 2$;

4) $d(5,2) = 0 + 0 = 0$;

5) $d(5,3) = 0 + 9 = 9$.

Наибольшая сумма констант приведения равна $(10 + 8) = 18$ для ребра $(2, 1)$. Включим ребро $(2, 1)$ в маршрут и удалим из матрицы C вторую строку и первый столбец. Получим матрицу 3×3 , которая подлежит операции приведения.

Сумма констант приведения сокращенной матрицы равна 0 ($H = 30$, т. е. ни один элемент не был изменен). Получим следующую матрицу:

	2	3	5
3	2	M	0
4	M	9	0
5	0	0	M

Чтобы исключить подциклы, запретим переход (4,2). Выполним приведение матрицы:

	2	3	5
3	2	M	0
4	M	9	0
5	0	0	M

Сумма констант приведения сокращенной матрицы равна 2. $H = 30 + 2 = 32$. Для всех клеток матрицы с нулевыми элементами заменим поочередно нули на « M » (бесконечность) и определим для них сумму образовавшихся констант приведения:

- 1) $d(3,5) = 2 + 0 = 2$;
- 2) $d(4,5) = 7 + 0 = 7$;
- 3) $d(5,2) = 0 + 2 = 2$;
- 4) $d(5,3) = 0 + 7 = 7$.

Наибольшая сумма констант приведения равна $(7 + 0) = 7$ для ребра (4,5) (выбрана первая максимальная сумма). Однако в данной точке алгоритм разветвляется, поэтому нужно рассмотреть все варианты выбора ребер.

1) Включение ребра (4,5) производится путем исключения всех элементов четвертой строки и пятого столбца. В результате получим матрицу 2×2 , которая подлежит операции приведения. Сумма констант приведения сокращенной матрицы равна 2 ($H = 30 + 2 = 32$), а матрица имеет вид:

	2	3
3	2	M
5	0	0

В пятой строке и втором столбце отсутствует элемент, равный « M ». Запретим переход (5,2). Тогда $H = 32$.

	2	3
3	2	M
5	M	0

Ранг этой матрицы равен двум. Получим нули в каждой строке и каждом столбце матрицы, а вычтенные элементы добавим к значению H . Тогда $H = 32 + 2 = 34$. В соответствии с последней матрицей включаем в маршрут ребра (3,2) и (5,3).

Итого маршрут коммивояжера включает в себя дуги: (1, 4), (2, 1), (4, 5), (3, 2), (5, 3), а длина этого маршрута равна 34.

2) Вернемся к ветвлению и рассмотрим случай, при котором максимальное значение имеет $d(5,3)$. Удалим из матрицы стоимости пятую строку и третий столбец. Внесем в текущий ориентированный граф дугу (5, 3). В результате получим матрицу 2×2 , которая подлежит операции приведения. Сумма констант приведения сокращенной матрицы равна 2 ($H = 30 + 2 = 32$), а матрица имеет вид:

	2	5
3	2	0
4	M	0

В третьей строке и пятом столбце отсутствует элемент, равный « M ». Присвоим элементу (3, 5) значение бесконечности, чтобы избежать преждевременного замыкания контура:

	2	5
3	2	M
4	M	0

Ранг этой матрицы равен двум. Получим нули в каждой строке и каждом столбце матрицы, вычтенные элементы добавим к значению H . Тогда $H = 32 + 2 = 34$. В соответствии с последней матрицей включим в маршрут ребра (3,2) и (4,5).

Итого маршрут коммивояжера включает в себя дуги: (1, 4), (2, 1), (5, 3), (3, 2), (4, 5), а длина этого маршрута равна 34.

Сравнив два полученных маршрута, можно увидеть, что они эквивалентны.

Ответ: (1, 4), (2, 1), (5, 3), (3, 2), (4, 5).

3.3.1. Приближенное решение уравнений

Существуют различные методы решения уравнений. Одним из них является приближенный метод. Его смысл заключается в определении отрезка числовой прямой (для уравнений с одной переменной), которому принадлежит корень уравнения, и в последующем уточнении значения корня. Для уточнения значения корня применяют *метод половинного деления, метод хорд и метод касательных*.

Метод половинного деления (деления отрезка пополам)

Метод половинного деления можно применять для уточнения значения корня уравнения, если функция $f(x)$ непрерывна на отрезке $[a; b]$ и на его концах принимает значения с разными знаками, т. е. если выполняется условие $f(a) \cdot f(b) < 0$.

Алгоритм метода половинного деления:

1) разделить отрезок $[a; b]$ пополам точкой $c = (a + b) / 2$, которая считается приближенным значением корня x' ;

2) из отрезков $[a; c]$ и $[c, b]$ выбрать тот, для которого выполняется неравенство $f(a) \cdot f(b) < 0$. Если это отрезок $[c, b]$, то $a = c$; если это отрезок $[a; c]$, то $b = c$. После этого выполняется переход вновь на шаг 1.

Шаги 1 и 2 нужно выполнять до тех пор, пока не станет истинным неравенство $|c_{i+1} - c_i| < \varepsilon$ (т. е. пока не будет достигнута требуемая точность вычисления корня).

Задача 12. Решить уравнение $x^3 + 7x + 11$ методом деления пополам с точностью до 0,01, если известно, что корень уравнения принадлежит отрезку $[-2; -1]$.

Решение.

Запишем по исходному уравнению функцию $f(x) = x^3 + 7x + 11$.

1. Найдем значения функции на концах отрезка:

$$f(-2) = (-2)^3 + 7 \cdot (-2) + 11 = -11, \quad f(-1) = (-1)^3 + 7 \cdot (-1) + 11 = 3.$$

2. Проверим выполнение неравенства:

$$f(-2) \cdot f(-1) = -11 \cdot 3 = -33 < 0.$$

Условие выполняется, следовательно, можно применить метод половинного деления.

3.1. Найдем середину отрезка $[-2; -1]$ и вычислим значение функции в полученной точке:

$$c = (-2 + (-1)) / 2 = -1,5, \quad f(-1,5) = -2,875 < 0.$$

3.2. Так как функция f принимает значения разных знаков в точках $f(-1,5)$ и $f(-1)$, то $a = -1,5$, и для дальнейшего анализа выбирается отрезок $[-1,5; -1]$.

4.1. Найдем середину отрезка $[-1,5; -1]$ и вычислим значение функции в полученной точке:

$$c = (-1,5 + (-1)) / 2 = -1,25, \quad f(-1,25) = 0,296 > 0.$$

4.2. Так как функция f принимает значения разных знаков в точках $f(-1,5)$ и $f(-1,25)$ и $|-1,5 - (-1,25)| = 0,25 > 0,01$, то $b = -1,25$ и для дальнейшего анализа выбирается отрезок $[-1,5; -1,25]$.

5.1. Найдем середину отрезка $[-1,5; -1,25]$ и вычислим значение функции в полученной точке:

$$c = (-1,5 + (-1,25)) / 2 = -1,375, \quad f(-1,375) = -1,224 < 0.$$

5.2. Так как функция f принимает значения разных знаков в точках $f(-1,375)$ и $f(-1,25)$ и $|-1,25 - (-1,375)| = 0,125 > 0,01$, то $a = -1,375$ и для дальнейшего анализа выбирается отрезок $[-1,375; -1,25]$.

6.1. Найдем середину отрезка $[-1,375; -1,25]$ и вычислим значение функции в полученной точке:

$$c = (-1,375 + (-1,25)) / 2 = -1,3125, \quad f(-1,3125) = -0,448 < 0.$$

6.2. Так как функция f принимает значения разных знаков в точках $f(-1,3125)$ и $f(-1,25)$ и $|-1,375 - (-1,3125)| = 0,0625 > 0,01$, то $a = -1,3125$, и для дальнейшего анализа выбирается отрезок $[-1,3125; -1,25]$.

7.1. Найдем середину отрезка $[-1,3125; -1,25]$ и вычислим значение функции в полученной точке:

$$c = (-1,3125 + (-1,25)) / 2 = -1,281, \quad f(-1,281) = -0,069 < 0.$$

7.2. Так как функция f принимает значения разных знаков в точках $f(-1,281)$ и $f(-1,25)$ и $|-1,3125 - (-1,281)| = 0,0315 > 0,01$, то $a = -1,281$, и для дальнейшего анализа выбирается отрезок $[-1,281; -1,25]$.

8.1. Найдем середину отрезка $[-1,281; -1,25]$ и вычислим значение функции в полученной точке:

$$c = (-1,281 + (-1,25)) / 2 = -1,265, \quad f(-1,265) = 0,120 > 0.$$

8.2. Так как функция f принимает значения разных знаков в точках $f(-1,281)$ и $f(-1,265)$ и $|-1,281 - (-1,265)| = 0,016 > 0,01$, то $b = -1,265$, и для дальнейшего анализа выбирается отрезок $[-1,281; -1,265]$.

9.1. Найдем середину отрезка $[-1,281; -1,265]$ и вычислим значение функции в полученной точке:

$$c = (-1,281 + (-1,265)) / 2 = -1,273, \quad f(-1,273) = 0,026 > 0.$$

9.2. Так как функция f принимает значения разных знаков в точках $f(-1,281)$ и $f(-1,273)$ и $|-1,281 - (-1,273)| = 0,008 < 0,01$, то достигнута заданная точность результата, и найдено приближенное значение корня $x' = -1,273$.

Ответ: $x' = -1,273$.

Метод хорд и касательных

Методы хорд и касательных можно использовать, если $f(a) \cdot f(b) < 0$ и $f'(x)$ и $f''(x)$ сохраняют знак на отрезке $[a; b]$.

Алгоритм метода хорд и касательных:

- 1) вычислить значения функции $f(a)$ и $f(b)$;
- 2) проверить выполнение условия $f(a) \cdot f(b) < 0$: если оно не выполняется, то неправильно выбран отрезок $[a; b]$;
- 3) вычислить первую и вторую производные $f'(x)$ и $f''(x)$;
- 4) проверить постоянство знака производных на отрезке $[a; b]$: если нет постоянства знака, то неверно выбран отрезок $[a; b]$;
- 5) для метода касательных за x_0 выбирается тот из концов отрезка $[a; b]$, для которого выполняется условие $f(x_0) \cdot f''(x_0) > 0$, т. е. для которого значения $f(x_0)$ и $f''(x_0)$ одного знака;
- 6) вычислить приближения корней:
 - по методу касательных: $x_1 = x_0 - f(x_0) / f'(x_0)$;
 - по методу хорд: $x_1 = a - ((b - a) \cdot f(a) / (f(b) - f(a)))$;
- 7) вычислить приближение корня: $\xi = (x_1 + x_2) / 2$;
- 8) проверить выполнение условия: $|\xi - x_1| < \varepsilon$, где ε — заданная точность. Если это условие не выполняется, то нужно продолжить выполнение шагов 1–8; при этом отрезок, на котором находится корень, сужается до $[x_1; x_2]$, $a = x_1$, $b = x_2$.

Задача 13. Решить уравнение $x^3 + 7x + 11 = 0$ методами хорд и касательных с точностью 0,1, если известно, что корень уравнения принадлежит отрезку $[-2; -1]$.

Решение.

1. Найдем значения функции на концах отрезка:

$$f(-2) = (-2)^3 + 7 \cdot (-2) + 11 = -11, \quad f(-1) = (-1)^3 + 7 \cdot (-1) + 11 = 3.$$

2. Проверим выполнение неравенства:

$$f(-2) \cdot f(-1) = -11 \cdot 3 = -33 < 0: \text{условие выполняется.}$$

3. Вычислим производные:

$$f'(x) = 3x^2 + 7, \quad f''(x) = 6x.$$

4. На отрезке $[-2; -1]$ производные $f'(x) > 0$ и $f''(x) < 0$, т. е. они сохраняют знак, следовательно, условие выполняется.

5. Выберем значение x_0 для метода касательных: $f''(-2) < 0$ и $f(-2) < 0$, поэтому $x_0 = -2$.

6. Найдем приближения корня:

- по методу касательных:

$$x_1 = -2 - f(-2) / f'(-2) = -2 - (-11) / 19 = -1,421;$$

- по методу хорд: $x_2 = -2 - (-1 - (-2)) \cdot f(-2) / (f(-1) - f(-2)) =$
 $= -2 - (-1 - (-2)) \cdot f(-2) / (f(-1) - f(-2)) =$
 $= -2 - 1 \cdot (-11) / (3 - (-11)) = -1,214.$

7. Вычислим приближение корня:

$$\xi = (-1,421 + (-1,214)) / 2 = -1,317.$$

8. Проверим выполнение условия $|\xi - x_1| = |-1,317 - (-1,421)| = 0,104 > 0,1$ — оно не выполняется, следовательно, нужно продолжить вычисления.

9. Новый отрезок, на котором находится корень:

$$x' = [-1,421; -1,214], a = -1,421, b = -1,214.$$

10. Найдем значения функции на концах отрезка:

$$f(-1,421) = -1,816, f(-1,214) = 0,712.$$

11. Условие $f(-1,421) \cdot f(-1,214) < 0$ выполняется, следовательно, можно продолжить применение метода.

12. Выберем значение x_0 для метода касательных: $f''(-1,421) < 0$ и $f(-1,421) < 0$, поэтому $x_0 = -1,421$.

13. Найдем приближения корня:

- по методу касательных: $x_1 = -1,421 - f(-1,421) / f'(-1,421) =$
 $= -1,421 - (-1,816) / 13,057 = -1,281;$

- по методу хорд: $x_2 = -1,421 - (-1,214 - (-1,421)) \cdot f(-1,421) /$
 $(f(-1,214) - f(-1,421)) = -1,421 - (-1,214 - (-1,421)) \cdot (-1,816) /$
 $(0,712 - (-1,816)) = -1,2723.$

14. Вычислим приближение корня: $\xi = (-1,281 + (-1,273)) / 2 = -1,277.$

15. Проверим выполнение условия: $|\xi - x_1| = |-1,277 - (-1,281)| = 0,004 < 0,1$ — оно выполняется, следовательно, $x' = -1,277.$

Ответ: $x' = -1,277.$

3.3.2. Поиски экстремумов функций

Метод равномерного поиска

Метод равномерного поиска экстремума функции $f(x)$ на отрезке $[a; b]$ основан на том, что переменной x присваиваются значения $x_i = x_0 + ih$ с шагом h , где $i = 0, 1, 2, \dots$, и вычисляются значения $f(x)$ в соседних точках. Если $f(x_{i+1}) \leq f(x_i)$, то переменной x дается новое

приращение. Поиск останавливается, если $f(x_{i+1}) > f(x_i)$, и предпоследняя точка считается ответом. Обычно в качестве x_0 выбирается один из концов отрезка.

Задача 14. Найти минимум функции $f(x) = 2x^2 - 12x$ методом равномерного поиска.

Решение.

График функции $f(x) = 2x^2 - 12x$ — парабола. Данная функция принимает нулевое значение в точках $x = 0$ и $x = 6$. Следовательно, минимум функции находится в интервале $(0; 6)$.

Пусть $x_0 = 0$, $h = 0,5$. Выполним поиск минимума функции.

1) $i = 1$, $x_1 = 0 + 1 \cdot 0,5 = 1$, $f(0,5) = -5,5$, $f(0) = 0$, $f(0,5) \leq f(0)$;

2) $i = 2$, $x_2 = 0 + 2 \cdot 0,5 = 1$, $f(1) = -10$, $f(1) \leq f(0,5)$;

3) $i = 3$, $x_3 = 0 + 3 \cdot 0,5 = 1,5$, $f(1,5) = -13,5$, $f(1,5) \leq f(1)$;

4) $i = 4$, $x_4 = 0 + 4 \cdot 0,5 = 2$, $f(2) = -16$, $f(2) \leq f(1,5)$;

5) $i = 5$, $x_5 = 0 + 5 \cdot 0,5 = 2,5$, $f(2,5) = -17,5$, $f(2,5) \leq f(2)$;

6) $i = 6$, $x_6 = 0 + 6 \cdot 0,5 = 3$, $f(3) = -18$, $f(3) \leq f(2,5)$;

7) $i = 7$, $x_7 = 0 + 7 \cdot 0,5 = 3,5$, $f(3,5) = -17,5$, $f(3,5) > f(3)$: минимум найден.

Ответ: $f(x) = \min$ при $x = 3$, $f(3) = -18$.

Метод деления отрезка пополам

Алгоритм поиска минимума $f(x)$ методом деления отрезка пополам состоит из следующих шагов:

1) пусть $x_m = (a + b) / 2$ и $L = b - a$, вычисляем значение $f(x_m)$;

2) пусть $x_1 = a + L / 4$ и $x_2 = b - L / 4$;

3) сравнить $f(x_1)$ и $f(x_m)$:

3.1) если $f(x_1) < f(x_m)$, то исключить интервал (x_m, b) , установить $b = x_m$, $x_m = x_1$ и перейти на шаг 5;

3.2) если $f(x_1) \geq f(x_m)$, то перейти на шаг 4;

4) сравнить $f(x_2)$ и $f(x_m)$:

4.1) если $f(x_2) < f(x_m)$, то исключить интервал (a, x_m) , установить $a = x_m$, $x_m = x_2$ и перейти на шаг 5;

4.2) если $f(x_2) \geq f(x_m)$, то исключить интервалы (a, x_1) и (x_2, b) , установить $a = x_1$ и $b = x_2$ и перейти на шаг 5;

5) $L = b - a$: если величина $|L|$ мала, то закончить поиск, иначе перейти на шаг 2.

Задача 13. Найти минимум функции $f(x) = (10 - 2x)^2$ в интервале $-20 \leq x \leq 20$ методом деления отрезка пополам.

Решение.

1. $a = -20, b = 20, L = 20 - (-20) = 40, x_m = (-20 + 20) / 2 = 0;$

$$x_1 = a + (L / 4) = -20 + (40 / 4) = -10;$$

$$x_2 = b - (L / 4) = 20 - (40 / 4) = 10;$$

$$f(-10) = 900 > f(0) = 100;$$

$$f(10) = 100 \leq f(0) = 100.$$

Таким образом, исключаются интервалы $(-20; -10)$ и $(10; 20)$.

2. $a = -10, b = 10, x_m = 0, L = 10 - (-10) = 20;$

$$x_1 = -10 + (20 / 4) = -5;$$

$$x_2 = 10 - (20 / 4) = 5;$$

$$f(-5) = 400 > f(0) = 100;$$

$$f(5) = 0 < f(0) = 10.$$

Минимальное значение функции равно 0, следовательно, на второй итерации алгоритма найден минимум функции в точке $x = 5$.

Ответ: $\min(f(5)) = 0$.

3.4. Принципы организации и функционирования вычислительных сетей

Вычислительная сеть — это совокупность компьютеров и телекоммуникационного оборудования, обеспечивающая обмен информацией между узлами сети. *Узел* — устройство, соединенное с другими устройствами сети. В качестве узла может выступать компьютер, маршрутизатор, коммутатор, концентратор, мобильный телефон, карманный компьютер, принтер и т. д. С каждым узлом сети связывается *MAC-адрес* — уникальный 48-битный идентификатор узла сети.

Вычислительные сети классифицируются по различным признакам, например по размеру охваченной территории, типу взаимодействия между узлами сети, сетевой технологии или по функциональному назначению.

Процесс обмена данными между узлами в сети определяется *сетевым протоколом* (набором правил). Для описания сетевых протоколов используется *сетевая модель OSI* — базовая эталонная модель взаимодействия открытых систем. В модели OSI взаимодействие между системами описывается на семи уровнях:

1) *физический уровень* предназначен для передачи электрических (оптических) сигналов по кабелю или в радиоэфир, их прием и преобразование в биты данных в соответствии с методами кодирования цифровых сигналов. К этому уровню имеют отношение характеристики физических сред передачи данных, такие как полоса пропускания, помехозащищенность, волновое сопротивление и др. На этом же уровне определяются характеристики электрических сигналов, стандартизируются типы разъемов и назначение каждого их контакта. На физическом уровне работают концентраторы, повторители сигнала и медиаконверторы (используются для соединения волоконно-оптических линии связи и витой пары);

2) *канальный уровень* обеспечивает формирование, передачу и прием кадров данных (*фреймов*) из битов данных, полученных с предыдущего уровня. Также на данном уровне выполняется коррекция ошибок во фреймах. На канальном уровне работают коммутаторы, мосты, драйверы сетевых карт;

3) *сетевой уровень* служит для объединения сетей с различными принципами передачи информации между конечными узлами. На этом уровне работает маршрутизатор;

4) *транспортный уровень* используется для обеспечения протоколам верхнего уровня передачи данных с той степенью надежности, которая им требуется;

5) *сеансовый уровень* управляет сеансами обмена данными, а именно фиксирует активную сторону и предоставляет средства синхронизации;

6) *представительский уровень* отвечает за преобразование протоколов и кодирование/декодирование данных. Запросы приложений, полученные с прикладного уровня, преобразуются в формат для передачи по сети, а полученные из сети данные — в формат, понятный приложениям;

7) *прикладной уровень* обеспечивает взаимодействие пользовательских приложений с сетью. Этот уровень позволяет приложениям использовать различные сетевые службы, например удаленный доступ к файлам и базам данных, пересылка электронной почты и т. д.

Часто при организации вычислительных сетей используется *семейство протоколов TCP/IP*. Стандарт TCP/IP был разработан в США в 1970-х гг. Название «TCP/IP» связано с двумя протоколами: TCP и IP. *Протокол IP* используется для негарантированной передачи данных между узлами сети (сетевой уровень модели).

Протокол TCP же функционирует на транспортном уровне модели OSI. Он выполняет предварительную установку соединения, за счет чего обеспечивается уверенность в достоверности получаемых данных. Протокол TCP также осуществляет повторный запрос данных в случае их потери и устраняет дублирование при получении двух копий одного пакета.

Каждый из узлов сети имеет уникальный *сетевой адрес*, который состоит из двух компонентов — *адреса сети (Network ID)* и *адреса узла (Host ID, IP-адрес)* в данной сети.

IP-адрес — адрес узла в сети, построенной по протоколу IP. Для обеспечения связи узлов требуется уникальность их адресов в пределах сети. IP-адреса для версии протокола IPv4 обычно представлены в виде четырех чисел, разделенных точками, например: **192.168.1.1**.

Для удобства анализа IP-адресов используют их двоичное представление в виде разрядов, разделенных точками. В данном представлении IP-адрес **192.168.123.132** имеет вид: **11000000.10101000.01111011.10000100**. Подобные последовательности двоичных символов длиной 8 элементов называются *октетами*.

IP-адрес состоит из двух частей. Первая часть IP-адреса обозначает адрес сети, а вторая — адрес узла. Для выделения этих частей используется *маска подсети*.

В терминологии сетей TCP/IP маской подсети (маской сети) называется битовая маска, определяющая, какая часть IP-адреса узла сети относится к адресу сети, а какая — к адресу узла в этой сети. Например, узел с IP-адресом **12.40.16.88** и маской подсети **255.255.0.0** находится в сети **12.40.0.0**. Двоичное представление этой маски подсети:

11111111.11111111.00000000.00000000.

Первые 16 разрядов (единиц в маске подсети) определяют адрес сети, а последние 16 разрядов (число оставшихся нулей в маске подсети) — адрес узла:

00001100.00101000.00000000.00000000 — адрес сети (**12.40.0.0**);

00000000.00000000.00010000.01011000 — адрес узла (**0.0.16.88**).

В каждой сети дополнительно резервируется два специальных адреса:

1) *адрес сети* — самый младший адрес сети (например, **128.8.0.0**). Адрес сети используется, когда требуется указать всю сеть целиком, скажем, при задании маршрута до сети или при определении, относится ли заданный узел к сети;

2) *широковещательный адрес (broadcast-адрес)* — самый старший адрес сети, резервируется для передачи сообщений всем узлам сети.

Возможно обращение к узлам сети не только по их IP-адресам, но и по *именам (host name)*. Список этих имен хранится в специальной базе данных *Domain Name System (DNS)*. Например, узлу с именем «*mail.ru*» в DNS соответствует IP-адрес **217.69.128.41**.

Связь между узлами из разных сетей обычно осуществляется через устройство, называемое *маршрутизатором*. С точки зрения TCP/IP, маршрутизатор, указанный на узле, связывающем подсеть узла с другими сетями, называется *основным шлюзом*.

При попытке установления связи между узлом и другим устройством с помощью протокола TCP/IP узел сопоставляет маску подсети и IP-адрес назначения с маской подсети и своим собственным IP-адресом. В результате становится известно, для какого узла предназначен пакет данных — для *локального* или для *удаленного*:

1) если данные должны быть отправлены на локальный узел, то компьютер просто отправляет пакет в локальную подсеть;

2) если пакет нужно отправить на удаленный узел, то компьютер направляет пакет на основной шлюз, определенный в свойствах протокола TCP/IP.

Таким образом, именно маршрутизатор отвечает за отправку пакета в правильную подсеть.

Самой большой вычислительной сетью, построенной на основе стандарта TCP/IP, является *Интернет (Internet* — сокращение от *Interconnected Networks*) — глобальная телекоммуникационная сеть информационных и вычислительных ресурсов. Для записи адреса ресурсов в сети Интернет используется *единый указатель ресурсов (URL — Uniform Resource Locator)*. Формат URL:

<схема>://<логин>:<пароль>@<узел>:<порт>/<URL-путь>,

где *схема* — схема обращения к ресурсу (в большинстве случаев имеется в виду сетевой протокол, например **ftp**, **http**, **https** и т. д.), *логин* — имя пользователя, используемое для доступа к ресурсу, *пароль* — пароль указанного пользователя, *узел* — доменное имя узла в системе DNS или его IP-адрес, *порт* — порт узла для подключения, *URL-путь* — уточняющая информация о местонахождении ресурса на диске узла (зависит от протокола).

Задача 1. Доступ к файлу **htm.net**, находящемуся на сервере **com.edu**, осуществляется по протоколу **ftp**. В таблице фрагменты адреса файла закодированы буквами от А до Ж.

А	/
Б	com
В	.edu
Г	://
Д	.net
Е	htm
Ж	ftp

Запишите последовательность этих букв, кодирующую адрес указанного файла в сети Интернет [4].

Решение.

При решении этой задачи используем формат URL, описанный выше. В тексте задания не указаны значения некоторых частей URL: *<логин>*, *<пароль>* и *<порт>*, поэтому мы будем рассматривать только составляющие, значения которых заданы в таблице:

1) доступ к файлу осуществляется по протоколу **ftp** (Ж), а после названия протокола записывается строка «://» (Г);

2) файл находится на сервере **com** (Б) **.edu** (В), а после имени сервера указывается наклонная черта «/» (А);

3) в качестве поля *<URL-имя>* используется имя файла **htm** (Е) **.net** (Д).

Таким образом, адрес файла можно закодировать с помощью следующей последовательности букв: ЖГБВАЕД.

Ответ: ЖГБВАЕД.

Задача 2. На месте преступления были обнаружены четыре обрывка бумаги. Следствие установило, что на них записаны фрагменты одного IP-адреса. Криминалисты обозначили эти фрагменты буквами А, Б, В и Г. Восстановите IP-адрес.

В ответе укажите последовательность букв, обозначающих фрагменты, в порядке, соответствующем IP-адресу [7].

.64	2.16	16	8.132
а	б	в	г

Решение.

На рис. 3.10 и рис. 3.11 показаны возможные адреса, которые можно построить из имеющихся фрагментов. Так как адрес с точки начинаться не может, то фрагмент «.64» не начинает ни один адрес. Кроме того, адреса на схемах, помеченные знаком «(–)», не могут

являться IP-адресами, так как десятичное значение никакой части адреса не должно превышать 255.

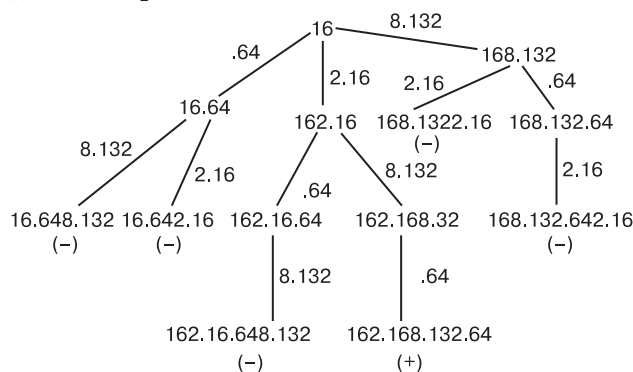


Рис. 3.10. Анализ фрагментов

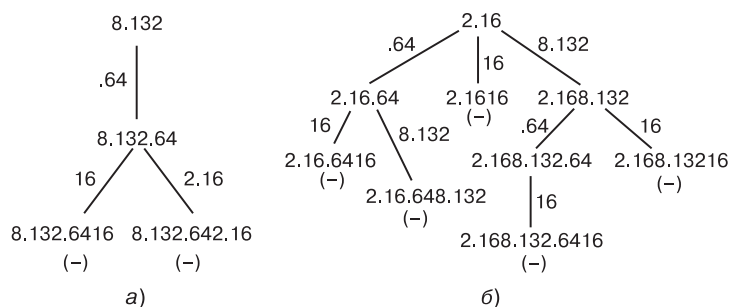


Рис. 3.11. Анализ фрагментов

Адрес, помеченный на схеме знаком «+», составлен из фрагментов, приведенных в тексте задания.

Ответ: вбга.

Задача 3. Определите, может ли значение 255.255.228.0 быть маской подсети.

Решение.

Для решения задачи получим двоичное представление маски. Если двоичное представление маски не представляет собой два набора идущих подряд единиц и нулей, то данный адрес маской быть не может.

255.255.228.0 = 11111111.11111111.11100100.00000000 — данное значение маской не является, так как набор единиц прерывается нулями (выделено подчеркиванием).

Ответ: приведенное значение не является маской подсети.

Задача 4. В таблице приведены запросы к поисковому серверу. Расположите номера запросов в порядке возрастания количества страниц, которые найдет поисковый сервер по каждому запросу. Для обозначения логической операции «ИЛИ» в запросе используется символ $|$, а для логической операции «И» — $\&$ [7].

№	Запрос
1	канарейки $ $ щеглы $ $ содержание
2	канарейки $\&$ содержание
3	канарейки $\&$ щеглы $\&$ содержание
4	разведение $\&$ содержание $\&$ канарейки $\&$ щеглы

Решение.

1-й способ:

Рассмотрим каждый из указанных в таблице запросов:

1) при поиске информации с помощью первого запроса будут выведены страницы, на которых находится хотя бы одно из слов: «канарейки», «щеглы», «содержания» или любые сочетания этих слов;

2) при использовании второго запроса круг страниц (по сравнению с первым) будет сокращен, так как между искомыми словами поставлен оператор $\&$. Использование данного оператора указывает на то, что страница будет включена в результат, только если на ней находится и слово «канарейки», и слово «содержание»;

3) в результате выполнения третьего запроса, по сравнению со вторым запросом, будет получено еще меньшее количество страниц, так как третьему запросу удовлетворяют лишь страницы, на которых одновременно находятся три слова: «канарейки», «щеглы» и «содержание»;

4) при использовании четвертого запроса будет получено еще меньше страниц от поискового сервера, так как этому запросу должны удовлетворять страницы, на которых одновременно находятся все четыре слова: «канарейки», «щеглы», «содержание», «разведение».

Таким образом, в порядке возрастания количества страниц запросы располагаются следующим образом: 4, 3, 2, 1.

2-й способ.

Для решения задачи можно воспользоваться *теорией множеств и кругами Эйлера*. Пусть X — множество страниц, выдаваемых по запросу «канарейки», Y — «щеглы», Z — «содержание», Q — «разведение». Тогда приведенные в тексте задания запросы с помощью кругов Эйлера представляются следующим образом (рис. 3.12).

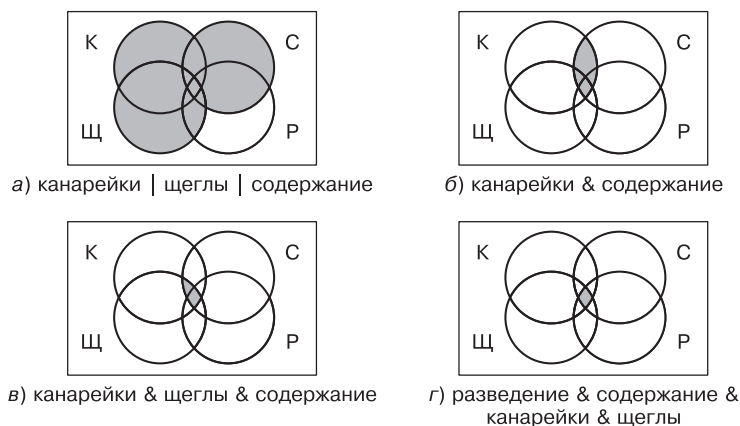


Рис. 3.12. Представление результатов выполнения запросов с помощью кругов Эйлера

Используя полученные рисунки, отсортируем запросы по увеличению числа страниц: 4, 3, 2, 1.

Ответ: 4, 3, 2, 1.

Задача 5. В таблице приведены запросы к поисковому серверу. Расположите обозначения запросов в порядке возрастания количества страниц, которые найдет поисковый сервер по каждому запросу. Если в запросе текст записан без кавычек, то производится поиск текстов, в которых встречаются все слова запроса. Если текст записан в кавычках, то производится поиск текстов, содержащих строго указанное словосочетание.

1	«Логическая операция»
2	«Логическая операция сумма по модулю два»
3	Логическая операция сумма по модулю два
4	Логическая операция

Решение.

В указанных запросах можно выделить две группы:

- 1) текст в запросе приведен в кавычках (1 и 2);
- 2) текст в запросе указан без кавычек (3 и 4).

Запросы второй группы накладывают менее строгие ограничения на страницу, которая удовлетворяет результатам поиска, так как для запросов первой группы порядок слов важен, а для второй — нет. Поэтому количество страниц, которые выдаст поисковый сервер при использовании запросов первой группы, меньше, чем при использовании запросов второй группы.

Рассмотрим теперь запросы отдельно в каждой группе.

При использовании первого запроса поисковый сервер должен найти страницы, на которых находится словосочетание «Логическая операция» (причем порядок слов важен). Второй запрос по сравнению с первым налагает более строгие ограничения на страницу, которая может быть включена в результат поиска. Поэтому при использовании первого поискового запроса будет выдано большее количество страниц, чем при использовании второго запроса.

При использовании третьего запроса по сравнению с четвертым будет выдано меньшее количество страниц, так как количество слов, используемых при поиске в третьем запросе, больше, чем в четвертом.

Таким образом, запросы в порядке возрастания количества страниц, которые найдет поисковый сервер, располагаются так: 2, 1, 3, 4.

Ответ: 2, 1, 3, 4.

Задача 6. Некоторый сегмент сети Интернет состоит из 1000 сайтов. Поисковый сервер в автоматическом режиме составил таблицу ключевых слов для сайтов этого сегмента. Вот ее фрагмент:

Ключевое слово	Количество сайтов, для которых данное слово является ключевым
Чемпион	500
России	300
Мира	150

Сколько сайтов будет найдено по запросу **Чемпион & (Мира | России)**, если по запросу **Мира | России** было найдено 450 сайтов, по запросу **Мира & Чемпион** — 13, а по запросу **России & Чемпион** — 41.

Решение.

Для решения задачи воспользуемся *теорией множеств*. Для отображения результатов поисковых запросов применим *круги Эйлера* (рис. 3.13).

Запишем для указанных запросов формулы:

Запрос	Формула	Номер формулы
Чемпион	$N_4 + N_5 + N_6 + N_7 = 500$	
России	$N_2 + N_3 + N_5 + N_6 = 300$	(3.8)
Мира	$N_1 + N_2 + N_4 + N_5 = 150$	(3.9)
Мира России	$N_1 + N_2 + N_3 + N_4 + N_5 + N_6 = 450$	(3.10)
Мира & Чемпион	$N_4 + N_5 = 13$	(3.11)
России & Чемпион	$N_5 + N_6 = 41$	(3.12)

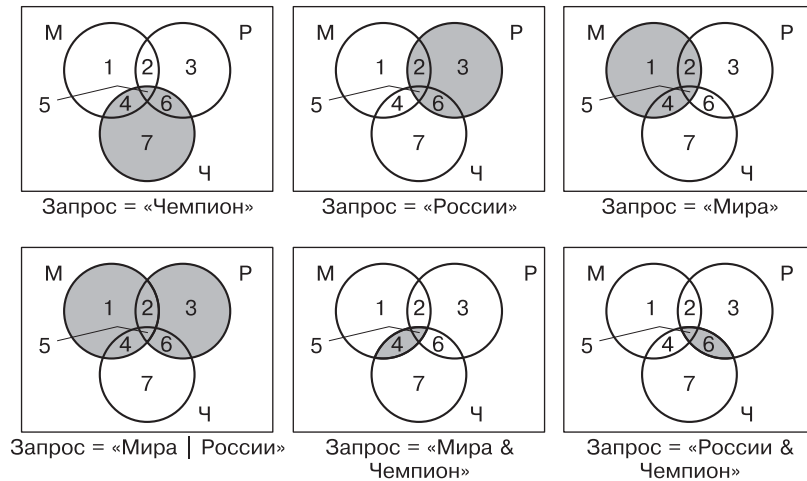
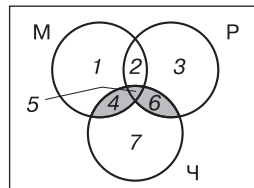


Рис. 3.13. Запросы и соответствующие им подмножества

На рис. 3.14 показано подмножество, формируемое в результате выполнения запроса **Чемпион & (Мира | России)**.

Рис. 3.14. Запрос = **Чемпион & (Мира | России)**

Таким образом, для решения задачи необходимо вычислить значение формулы $N_4 + N_5 + N_6$.

Подставим (3.9) в (3.10), получим:

$$N_3 + N_6 = 300. \quad (3.13)$$

Изучив формулы (3.13) и (3.8), можно сделать вывод, что $N_2 + N_5 = 0$, так как $N_2 + N_3 + N_5 + N_6 = 300$ и $N_3 + N_6 = 300$. Следовательно, $N_2 = 0$, $N_5 = 0$.

Подставляя N_5 в (3.11) и (3.12), вычислим значения N_4 и N_6 : $N_4 = 13$, $N_6 = 41$.

Таким образом, $N_4 + N_5 + N_6 = 54$. Следовательно, запросу **Чемпион & (Мира | России)** удовлетворяют 54 записи.

Ответ: 54.

3.5. Организация файловых систем

Файловая система — стандарт, описывающий способ хранения и организацию доступа к данным на том или ином носителе или его разделе. *Раздел* (также иногда называемый томом) — это область носителя информации, которая может быть отформатирована под определенную файловую систему. В операционных системах семейства Windows раздел обозначается одной латинской буквой. На одном носителе информации, например на жестком диске, может находиться несколько разделов.

Файл — поименованная совокупность областей данных на носителе информации.

Расширение имени файла — последовательность символов, добавляемых к имени файла и предназначенных для идентификации типа файла. Расширение обычно отделяется от основной части имени файла точкой. Необходимо учитывать, что данные в файле могут не соответствовать заявленному расширению. В некоторых операционных системах, например в Linux, расширение имени файла может отсутствовать.

Многие современные файловые системы в качестве основного организационного средства используют каталоги. *Каталог* — это список ссылок на файлы или на другие каталоги. (Принято говорить, что каталог *содержит* в себе файлы или другие каталоги, хотя в действительности он только *ссылается* на них. Физическое размещение данных на диске при этом обычно никак не связано с размещением каталога.) Каталог, на который есть ссылка в данном каталоге, называется *подкаталогом* или *вложенным каталогом*.

В каждом каталоге также предусмотрены два специальных элемента:

- 1) «.» (точка) — обозначает сам этот (текущий) каталог;
- 2) «..» (две точки) — ссылка на *родительский каталог* для текущего.

Допустимые действия над файлами:

- 1) *копирование* — копия файла помещается в другой каталог, но исходный файл остается на прежнем месте;
- 2) *перемещение* — файл перемещается в другой каталог и удаляется из прежнего места;
- 3) *удаление* — запись о файле удаляется из каталога;
- 4) *переименование* — изменяется имя файла;
- 5) *редактирование* — получение доступа к данным, хранящимся в файле.

При выполнении различных действий с *группами файлов* используется *маска имен файлов*. Такая маска представляет собой последовательность букв, цифр и прочих допустимых в именах файлов символов, среди которых также могут встречаться следующие специальные символы:

1) «?» (вопросительный знак) — заменяет ровно один произвольный символ;

2) «*» (звездочка) — заменяет любую последовательность символов произвольной длины, в том числе пустую последовательность.

Например, маска ***test.??** соответствует именам файлов **test.01** и **000test.02**, но не соответствует именам **test.001** или **000test.2**.

Уникальность имени файла (либо каталога) обеспечивается тем, что *полным именем файла (каталога)* считается собственное имя файла (каталога) вместе с путем доступа к нему. *Путь доступа к файлу (каталогу)* начинается с имени устройства и включает все имена каталогов, через которые должен проходить пользователь для получения доступа к файлу (каталогу). В качестве разделителя в операционных системах семейства Windows используется символ «\» (обратная косая черта), например: **C:\temp\setup.exe**.

Файловая система может быть организована в виде дерева или сети (рис. 3.15). Каталоги образуют *дерево*, если файлу разрешено входить только в один каталог, или *сеть* — если файл может входить сразу в несколько каталогов.

Каталог, находящийся на вершине иерархии, называется *корневым*.

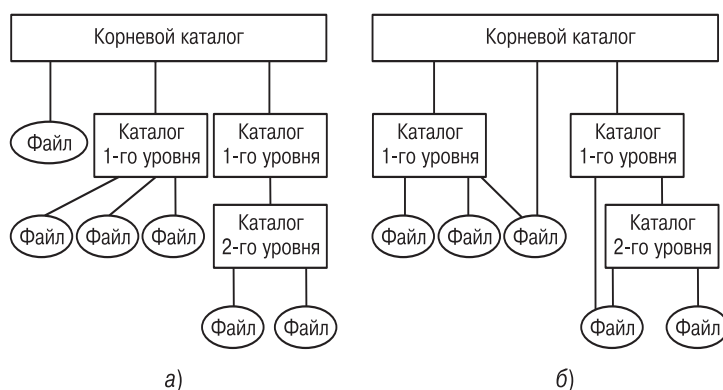


Рис. 3.15. Логическая организация файловой системы: а) дерево; б) сеть

Задача 1. Определите, удовлетворяет ли имя файла `barr.txt` маске: `?ba*r.?xt` [7].

Решение.

Маска `?ba*r.?xt` определяет следующие требования к имени файла:

- 1) второй и третий символы имени — **ba**;
- 2) имя файла должно заканчиваться на **xt**;
- 3) между символом **.** (точка) и **xt** должен находиться ровно один символ.

Имя `barr.txt` не соответствует первому требованию, так как второй и третий символы — **ar**, а нужно — **ba**.

Ответ: не удовлетворяет.

Задача 2. Определите, удовлетворяет ли имя файла `dad22` маске `?a****` [5].

Решение.

Маска `?a****` определяет следующие требования к имени файла:

- 1) второй символ в имени файла — буква **a**;
- 2) имя файла должно содержать как минимум 5 символов (так как символ «*****» может задавать пустую последовательность).

Имя `dad22` отвечает обоим требованиям (длина имени — 5 символов, второй символ — **a**).

Ответ: удовлетворяет.

Задача 3. Перемещаясь из одного каталога в другой, пользователь последовательно посетил каталоги **DOC**, **USER**, **SCHOOL**, **A:**, **LETTER**, **INBOX**. При каждом перемещении пользователь либо спускался в каталог на уровень ниже, либо поднимался на уровень выше. Каково полное имя каталога, из которого начал перемещение пользователь [4]?

Решение.

Пользователь начал перемещение из каталога **DOC**. Каталог **A:** является корневым каталогом раздела **A**. Достичь каталога **A:** из каталога **DOC** можно, только поднимаясь на уровень вверх. Поэтому полное имя каталога **DOC** будет включать все каталоги, которые посетил пользователь при перемещении из **DOC** в **A:**, а именно: **A:\SCHOOL\USER\DOC** (рис. 3.16).

Ответ: **A:\SCHOOL\USER\DOC**.

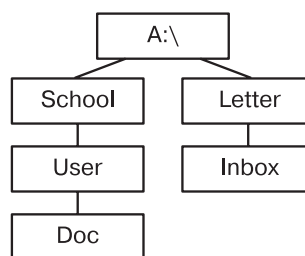


Рис. 3.16. Дерево каталогов

Задача 4. В некотором каталоге хранился файл **Задача5**. После того как в этом каталоге создали подкаталог и переместили в созданный подкаталог файл **Задача5**, полное имя файла стало **Е:\Класс9\Физика\Задачник\Задача5**. Каково было полное имя этого файла до его перемещения [2]?

Решение.

Файл **Задача5** теперь находится в подкаталоге **Задачник**. Следовательно, именно этот подкаталог и был создан. Подкаталог **Задачник** находится в каталоге **Физика**, следовательно, файл **Задача5** до перемещения находился в каталоге **Физика**. Поэтому полное имя файла **Задача5** до его перемещения было: **Е:\Класс9\Физика\Задача5**.

Ответ: **Е:\Класс9\Физика\Задача5**.

Задача 5. Каталог содержит файлы

а) **p1.roc**; б) **p21.doc**; в) **p4.d**; г) **p33.d**; д) **pap.roc**; е) **pom.pd**.

Определите список всех выделенных файлов при их выделении с использованием маски **p*.p??**.

Решение.

Файлы, соответствующие указанной маске, должны отвечать следующим требованиям:

1) первая буква имени — **p**;

2) после буквы **p** в расширении имени файла обязательно должно находиться ровно два символа.

Указанным требованиям отвечают имена файлов: а) **p1.roc**; д) **pap.roc**.

Ответ: а, д.

Задача 6. Каталог содержит файлы:

а) **p1.doc**; б) **p21.doc**; в) **p4.d**; г) **p33.d**; д) **pap.doc**; е) **pom.dd**.

В какой последовательности будут представлены эти файлы после их упорядочивания по типу (по возрастанию)?

Решение.

При упорядочивании файлов по типу сравниваются коды символов в расширениях имен файлов, а при сравнении расширений разной длины считается, что расширение с меньшей длиной дописывается символами с нулевым кодом. Поэтому при сравнении, например, расширений **dd** и **doc** в случае сортировки по возрастанию первым окажется файл с расширением **dd**, а следующим — с расширением **doc**. Если же у двух сравниваемых файлов окажутся одинаковые расширения, то выполняется сравнение имен файлов.

Максимальная длина расширения файлов в указанном списке — три символа. Соответственно, все расширения с меньшей длиной будут дополнены до трех знаков символами с нулевым кодом. В результате в списке отсортированных файлов сначала будут расположены файлы с расширением **d**, потом — с расширением **dd**, а затем с расширением **doc**, так как код символа **d** меньше кода символа **o**.

Рассмотрим теперь, как будут располагаться в списке файлы с одинаковым расширением:

1) порядок следования файлов с расширением **d**: сначала будет идти файл **p33.d**, а затем — файл **p4.d**, так как код символа **3** меньше кода символа **4**;

2) порядок следования файлов с расширением **doc**: **p1.doc**, **p21.doc**, **pap.doc**, так как код символа **2** больше кода символа **1**, а код символа **a** больше кода символа **2**.

Ответ: г, в, е, а, б, д.

Задачи для самостоятельного решения

Моделирование

Задача 1. Путешественник пришел в 08:00 на автостанцию населенного пункта ДРУЖБА и обнаружил следующее расписание автобусов:

Пункт отправления	Пункт прибытия	Время отправления	Время прибытия
ДРУЖБА	ВЫСОКОВО	08:00	08:15
ДРУЖБА	КИРПИЧНЫЙ	08:05	09:05
ВЫСОКОВО	КИРПИЧНЫЙ	08:15	09:00
ВЫСОКОВО	ДРУЖБА	08:35	09:00
ВЫСОКОВО	СОВОЛИХА	08:40	08:50
КИРПИЧНЫЙ	СОВОЛИХА	09:05	09:20
КИРПИЧНЫЙ	ДРУЖБА	09:10	10:10
СОВОЛИХА	КИРПИЧНЫЙ	09:20	09:35

Определите самое раннее время, когда путешественник сможет оказаться в пункте КИРПИЧНЫЙ согласно этому расписанию.

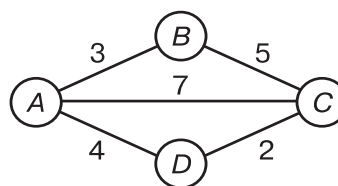
Задача 2. Путешественник пришел в 08:00 на автостанцию населенного пункта ЗЕМЛЯНОЕ и обнаружил следующее расписание автобусов:

Пункт отправления	Пункт прибытия	Время отправления	Время прибытия
ЗЕМЛЯНОЕ	ЯГДНОЕ	08:00	09:00
СТАРЫЕ ТОПОЛЯ	КАЛИНИНО	09:05	09:25
СТАРЫЕ ТОПОЛЯ	ЗЕМЛЯНОЕ	09:10	09:45
ЯГДНОЕ	ЗЕМЛЯНОЕ	10:00	11:00
ЯГДНОЕ	СТАРЫЕ ТОПОЛЯ	10:05	10:55
ЯГДНОЕ	КАЛИНИНО	10:15	10:25
КАЛИНИНО	СТАРЫЕ ТОПОЛЯ	10:15	10:35
ЗЕМЛЯНОЕ	СТАРЫЕ ТОПОЛЯ	10:45	11:20

Определите самое раннее время, когда путешественник сможет оказаться в пункте СТАРЫЕ ТОПОЛЯ согласно этому расписанию.

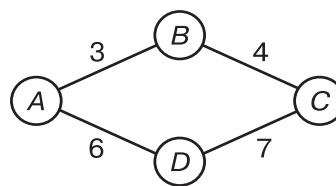
Задача 3. В таблице приведена стоимость перевозок между соседними железнодорожными станциями. Определите, соответствует ли приведенная ниже схема таблице:

	A	B	C	D
A		3	7	4
B	3		5	
C	7	5		2
D	4		2	



Задача 4. В таблице приведена стоимость перевозок между соседними железнодорожными станциями. Определите, соответствует ли приведенная ниже схема таблице:

	A	B	C	D
A		3		
B	3		4	6
C		4		7
D		6	7	



Задача 5. Таблица стоимости перевозок устроена следующим образом: числа, стоящие на пересечениях строк и столбцов, обозначают стоимость проезда между соответствующими соседними станциями. Если пересечение строки и столбца пусто, то станции не являются соседними. Стоимость про-

	A	B	C	D	E
A			7	1	
B			4		3
C	7	4			
D	1				5
E		3		5	

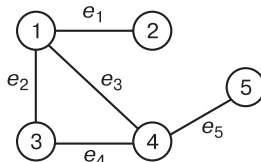
езда по маршруту складывается из стоимостей проезда между соответствующими соседними станциями.

Определите, выполняется ли для приведенной таблицы условие: «Минимальная стоимость проезда из A в B меньше 10».

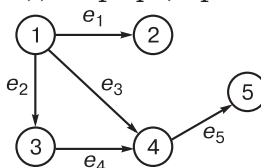
Задача 6. Грунтовая дорога проходит последовательно через населенные пункты A , B , C и D . При этом длина дороги между A и B равна 40 км, между B и C — 30 км, и между C и D — 50 км. Между B и D построили новое асфальтовое шоссе длиной 70 км. Оцените минимально возможное время движения велосипедиста из пункта A в пункт C , если его скорость по грунтовой дороге — 20 км/ч, а по шоссе — 50 км/ч.

Задача 7. Грунтовая дорога проходит через населенные пункты A , B , C и D . При этом длина дороги между A и B равна 10 км, между B и D — 80 км, а между B и C — 20 км. Между C и D построили новое асфальтовое шоссе длиной 80 км. Оцените минимально возможное время движения велосипедиста из пункта A в пункт D , если его скорость по грунтовой дороге — 20 км/ч, а по шоссе — 40 км/ч.

Задача 8. Постройте матрицу смежности, матрицу инцидентности и списки пар вершин для графа, приведенного на рисунке:



Задача 9. Постройте матрицу смежности, матрицу инцидентности и списки пар вершин для графа, приведенного на рисунке:



Задача 10. Постройте неориентированный граф по матрице смежности:

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

Задача 11. Постройте неориентированный граф по матрице инцидентности:

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

Задача 12. Постройте ориентированный граф по матрице смежности:

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

Задача 13. Постройте ориентированный граф по матрице инцидентности:

1	-1	0	0	0	0	0
0	1	-1	-1	0	0	0
0	0	1	0	-1	0	0
-1	0	0	0	0	-1	0
0	0	0	1	0	1	-1
0	0	0	0	1	0	1

Задача 14. Постройте неориентированный граф по списку вершин: $\{1, 2\}, \{1, 3\}, \{2, 3\}, \{2, 4\}, \{2, 5\}, \{4, 5\}$.

Задача 15. Постройте ориентированный граф по списку вершин: $((1, 5), (2, 1), (3, 1), (5, 2), (5, 3), (5, 4))$.

Задача 16. Найдите максимум функции $f(x) = -2x^2 - 2x$ методом равномерного поиска.

Задача 17. Найдите максимум функции $f(x) = -3x^2 - 5x + 6$ методом равномерного поиска.

Задача 18. Найдите минимум функции $f(x) = 7x^2 + 6$ методом равномерного поиска.

Задача 19. Найдите максимум функции $f(x) = -7x^2 + 5$ методом половинного деления на отрезке $[-4; 10]$.

Задача 20. Найдите минимум функции $f(x) = 2x^2 + 5$ методом половинного деления на отрезке $[-7; 10]$.

Задача 21. Найдите максимум функции $f(x) = -3x^2 + 1$ методом половинного деления на отрезке $[-3; 10]$.

Задача 22. Решите уравнение $x^3 + 3x - 8 = 0$ методом хорд и касательных с точностью 0,01, если известно, что корень уравнения принадлежит отрезку $[1; 2]$.

Задача 23. Решите уравнение $x^3 + 5x - 10 = 0$ методом хорд и касательных с точностью 0,01, если известно, что корень уравнения принадлежит отрезку $[1; 2]$.

Задача 24. Решите уравнение $x^3 - 2x - 12 = 0$ методом хорд и касательных с точностью 0,01, если известно, что корень уравнения принадлежит отрезку $[2; 3]$.

Задача 25. Решите уравнение $x^3 + 4x + 6 = 0$ методом хорд и касательных с точностью 0,01, если известно, что корень уравнения принадлежит отрезку $[-2; -1]$.

Задача 26. Решите уравнение $x^3 - 1x - 7 = 0$ методом деления отрезка пополам с точностью до 0,01, если известно, что корень уравнения принадлежит отрезку $[2; 3]$.

Задача 27. Решите уравнение $x^3 + 9x + 4 = 0$ методом деления отрезка пополам с точностью до 0,01, если известно, что корень уравнения принадлежит отрезку $[-1; 0]$.

Задача 28. Решите уравнение $x^3 + 9x - 2 = 0$ методом деления отрезка пополам с точностью до 0,01, если известно, что корень уравнения принадлежит отрезку $[1; 2]$.

Задача 29. Решите уравнение $x^3 - 3x + 10 = 0$ методом деления отрезка пополам с точностью до 0,01, если известно, что корень уравнения принадлежит отрезку $[-3; -2]$.

Задача 30. Решите задачу линейного программирования симплексным методом с использованием симплексной таблицы. Определите максимальное значение целевой функции $f(x_1, x_2) = x_1 + x_2$ при следующих ограничениях:

$$x_1 + x_2 \leq 1, \quad x_1 - x_2 \leq 1.$$

Задача 31. Решите задачу линейного программирования симплексным методом, с использованием симплексной таблицы. Определите минимальное значение целевой функции $f(x_1, x_2) = 7x_1 + 5x_2$ при следующих ограничениях:

$$x_1 + x_2 \geq 3; \quad x_1 + 5x_2 \leq 5; \quad 2x_1 + x_2 \leq 4.$$

Задача 32. Решите задачу линейного программирования табличным симплекс-методом. Определите максимальное значение целевой функции $f(x_1, x_2) = 5x_1 + 3x_2$ при следующих ограничениях:

$$x_1 + 2x_2 \leq 4; \quad 5x_1 + 2x_2 \leq 10.$$

Задача 33. Решите задачу линейного программирования табличным симплекс-методом. Определите минимальное значение целевой функции $f(x_1, x_2) = x_1 + 2x_2$ при следующих ограничениях:

$$x_1 - x_2 \geq 1; \quad x_1 - 2x_2 \geq 1.$$

Задача 34. Решите задачу линейного программирования табличным симплекс-методом. Определите минимальное значение целевой функции $f(x_1, x_2) = -x_1 + 2x_2$ при следующих ограничениях:

$$2x_1 - 3x_2 \geq 0; \quad x_1 - x_2 \leq 3; \quad 2x_1 - x_2 \geq 4.$$

Задача 35. Решите задачу линейного программирования табличным симплекс-методом. Определите минимальное значение целевой функции $f(x_1, x_2) = x_1 + x_2$ при следующих ограничениях:

$$x_1 + x_2 \geq 1; \quad x_1 - x_2 \geq 1.$$

Задача 36. Коммивояжер должен побывать в 5 городах. Для сокращения расходов он должен выбрать такой маршрут, чтобы посетить каждый город только один раз и вернуться в исходный с минимумом затрат. Исходный город — 1. Затраты на перемещение между городами заданы матрицей:

	1	2	3	4	5
1	М	3	4	3	7
2	1	М	12	6	8
3	10	1	М	7	6
4	5	7	1	М	6
5	2	3	4	10	М

Решите задачу методом ветвей и границ.

Задача 37. Коммивояжер должен побывать в 5 городах. Для сокращения расходов он должен выбрать такой маршрут, чтобы посетить каждый город только один раз и вернуться в исходный с минимумом затрат. Исходный город — 1. Затраты на перемещение между городами заданы матрицей:

	1	2	3	4	5
1	М	7	8	2	5
2	6	М	5	5	3
3	1	3	М	5	4
4	6	3	1	М	8
5	1	2	7	9	М

Решите задачу методом ветвей и границ.

Задача 38. Коммивояжер должен побывать в 4 городах. Для сокращения расходов он должен выбрать такой маршрут, чтобы посетить каждый город только один раз и вернуться в исходный с минимумом затрат. Исходный город — 1. Затраты на перемещение между городами заданы матрицей:

	1	2	3	4
1	M	12	11	10
2	9	M	5	5
3	6	7	M	9
4	8	2	1	M

Решите задачу методом ветвей и границ.

Задача 39. Коммивояжер должен побывать в 4 городах. Для сокращения расходов он должен выбрать такой маршрут, чтобы посетить каждый город только один раз и вернуться в исходный с минимумом затрат. Исходный город — 1. Затраты на перемещение между городами заданы матрицей:

	1	2	3	4
1	M	3	5	2
2	4	M	6	5
3	2	3	M	1
4	5	4	7	M

Решите задачу методом ветвей и границ.

Задача 40. Коммивояжер должен побывать в 4 городах. Для сокращения расходов он должен выбрать такой маршрут, чтобы посетить каждый город только один раз и вернуться в исходный с минимумом затрат. Исходный город — 1. Затраты на перемещение между городами заданы матрицей:

	1	2	3	4
1	M	12	8	4
2	6	M	5	6
3	1	3	M	8
4	7	2	3	M

Решите задачу методом ветвей и границ.

Задача 41. Коммивояжер должен побывать в 4 городах. Для сокращения расходов он должен выбрать такой маршрут, чтобы посетить каждый город только один раз и вернуться в исходный с минимумом затрат. Исходный город — 1. Затраты на перемещение между городами заданы матрицей:

	1	2	3	4
1	М	5	10	7
2	8	М	6	5
3	4	3	М	9
4	6	5	4	М

Решите задачу методом ветвей и границ.

Задача 42. Решите задачу линейного программирования графическим способом:

$$f(x_1, x_2) = 2x_1 + 3x_2 \rightarrow \max, x_1 + x_2 \leq 6, x_1 + 2x_2 \leq 4, 2x_1 + x_2 \leq 5, x_1 \geq 0, x_2 \geq 0.$$

Задача 43. Решите задачу линейного программирования графическим способом:

$$f(x_1, x_2) = 2x_1 + 3x_2 \rightarrow \min, x_1 + x_2 \geq 6, x_1 + 2x_2 \geq 4, 2x_1 + x_2 \geq 5, x_1 \geq 0, x_2 \geq 0.$$

Задача 44. Решите задачу линейного программирования графическим способом:

$$f(x_1, x_2) = 2x_1 + x_2 \rightarrow \min, x_1 + x_2 \leq 2, x_1 + 2x_2 \leq 7, 4x_1 + 3x_2 \leq 6, x_1 \geq 0, x_2 \geq 0.$$

Задача 45. Решите задачу линейного программирования графическим способом:

$$f(x_1, x_2) = 3x_1 + 2x_2 \rightarrow \max, x_1 + 2x_2 \leq 6, 2x_1 + x_2 \leq 8, -x_1 + x_2 \leq 1, x_1 \geq 0, x_2 \geq 0.$$

Принципы организации и функционирования вычислительных сетей

Задача 1. На сервере **mat.sc** находится файл **document.txt**, доступ к которому осуществляется по протоколу **ftp**. Фрагменты адреса закодированы буквами *a, b, c* и т. д. Запишите последовательность этих букв, которая кодирует адрес указанного файла в Интернете.

<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>	<i>g</i>
mat	document	://	.txt	ftp	.sc	/

Задача 2. На сервере **city.nn** находится файл **document.pdf**, доступ к которому осуществляется по протоколу **http**. Фрагменты адреса закодированы буквами *a, b, c* и т. д. Запишите последовательность этих букв, которая кодирует адрес указанного файла в Интернете.

<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>	<i>g</i>
city	document	://	.pdf	http	.nn	/

Задача 3. Были найдены четыре фрагмента документа. Было установлено, что на них записаны фрагменты одного IP-адреса. Эти фрагменты были обозначены буквами *a*, *b*, *c* и *d*. Восстановите IP-адрес. В ответе укажите последовательность букв, обозначающих фрагменты, в порядке, соответствующем IP-адресу.

<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>
9.25	.248	18	89.

Задача 4. Файл с именем **file**, созданный в текстовом редакторе «Блокнот», скачивается по протоколу передачи файлов с интернет-узла с домена второго уровня **dec**, расположенного в коммерческих сетях США. Известно, что в URL скачиваемого файла не указано никаких других доменов. Выберите из таблицы только необходимые фрагменты адреса файла и запишите последовательность букв, кодирующих этот адрес в сети Интернет.

<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>	<i>g</i>	<i>h</i>	<i>i</i>	<i>j</i>
http	ftp	www.	.com	.ru	://	/	.dec	file	.txt

Задача 5. Были найдены четыре фрагмента документа. Было установлено, что на них записаны фрагменты одного IP-адреса. Эти фрагменты были обозначены буквами *a*, *b*, *c* и *d*. Восстановите IP-адрес. В ответе укажите последовательность букв, обозначающих фрагменты, в порядке, соответствующем IP-адресу.

<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>
1.14	30	3.110.	20

Задача 6. Ниже приведены запросы к поисковому серверу. Расположите обозначения этих запросов в порядке возрастания количества страниц, которые найдет поисковый сервер по каждому запросу:

- Торпедо | СКА;
- СКА & Торпедо & Динамо;
- Торпедо & СКА;
- СКА & Торпедо & Динамо & Спартак.

Задача 7. Ниже приведены запросы к поисковому серверу. Расположите обозначения этих запросов в порядке убывания количества страниц, которые найдет поисковый сервер по каждому запросу:

- Комплектация & ОЗУ;
- (ПЗУ | ОЗУ) & Комплектация;
- (ПЗУ | ОЗУ) | Комплектация;
- (ПЗУ & ОЗУ & Процессор) & Комплектация.

Задача 8. Определите, может ли значение **255.205.203.0** быть маской подсети.

Задача 9. Определите, может ли значение **255.128.0.0** быть маской подсети.

Задача 10. Сервер, на котором предоставляются услуги размещения сайтов, содержит определенное количество сайтов. Поисковый сервер в автоматическом режиме составил таблицу ключевых слов для сайтов данного сервера. Фрагмент этой таблицы:

Ключевое слово	Количество сайтов сервера, для которых данное слово является ключевым
F1	250
Петров	200
Рено	500

Сколько сайтов будет найдено по запросу **(F1 & Петров) | Рено**, если по запросу **F1 | Рено** было найдено 750 сайтов, по запросу **Петров & F1** — 170, а по запросу **Петров & Рено** — 0.

Задача 11. Сервер, на котором предоставляются услуги размещения сайтов, содержит определенное количество сайтов. Поисковый сервер в автоматическом режиме составил таблицу ключевых слов для сайтов данного сервера. Фрагмент этой таблицы:

Ключевое слово	Количество сайтов, для которых данное слово является ключевым
Динамо Р	200
Торпедо	250
КХЛ	450

Сколько сайтов будет найдено по запросу **(Динамо Р & Торпедо) | КХЛ**, если по запросу **Динамо Р | Торпедо** было найдено 450 сайтов, по запросу **Торпедо & КХЛ** — 40, а по запросу **Динамо Р & КХЛ** — 50.

Задача 12. Сервер, на котором предоставляются услуги размещения сайтов, содержит определенное количество сайтов. Поисковый сервер в автоматическом режиме составил таблицу ключевых слов для сайтов данного сервера. Фрагмент этой таблицы:

Ключевое слово	Количество сайтов, для которых данное слово является ключевым
Олимпиада	200
Россия	350
Гагарин	200

Сколько сайтов будет найдено по запросу **Олимпиада | Россия | Гагарин**, если по запросу **Олимпиада | Гагарин** было найдено 0 сайтов, по запросу **Олимпиада & Россия** — 99, а по запросу **Гагарин & Россия** — 35.

Обработка информации в электронных таблицах

Задача 1. Дан фрагмент электронной таблицы:

	A	B	C
1	2	3	=2*A1 + 3*B\$1
2	7	4	

Чему станет равным значение ячейки **C2**, если в нее скопировать формулу из ячейки **C1**? Знак \$ обозначает абсолютную адресацию.

Задача 2. Дан фрагмент электронной таблицы:

	A	B	C
1	2	5	=A\$1^2 + B1 + 3
2	3	7	

Чему станет равным значение ячейки **C2**, если в нее скопировать формулу из ячейки **C1**? Знак \$ обозначает абсолютную адресацию.

Задача 3. Дан фрагмент электронной таблицы:

	A	B	C	D
1		Стоимость литра топлива (руб)	22	
2	Марка автомобиля	Пробег за день (км)	Расход топлива (л / 100 км)	Стоимость (руб.)
3	ВАЗ 2107	20	8	
4	ГАЗ 3102	30	10	
5	ВАЗ 2108	30	7	

Какая формула должна быть введена в ячейку **D3** для последующего автозаполнения нижестоящих значений?

Задача 4. Дан фрагмент электронной таблицы:

	A	B	C	D
1	Стоимость литра топлива (руб.)	22	Средний пробег за день	120
2	Марка автомобиля	Расход топлива (л / 100 км)	Стоимость (руб.)	
3	ВАЗ 2107	8		
4	ГАЗ 3102	10		
5	ВАЗ 2108	7		

Какая формула должна быть введена в ячейку **C3** для последующего автозаполнения нижестоящих значений?

Задача 5. Представлен фрагмент электронной таблицы, содержащий числа и формулы:

	B	C	D
69	3	2	
70	7	9	=МАКС(B69:C70)
71			=СУММ(B69:D70)

Определите, как изменится по абсолютной величине значение в ячейке **D71** после перемещения содержимого ячейки **C70** в ячейку **C71**.

Задача 6. Представлен фрагмент электронной таблицы, содержащий числа и формулы:

	B	C	D
69	5	8	
70	7	10	=МАКС(B69:C70)
71	14	15	=СРЗНАЧ (B69:D70)

Определите, как изменится по абсолютной величине значение в ячейке **D71** после перемещения содержимого ячейки **C70** в ячейку **C71**.

Задача 7. Представлен фрагмент электронной таблицы, содержащий числа и формулы:

	B	C	D
69	3	8	=МИН(B69:C70)
70	7	2	=МАКС(B69:C70)
71	3	1	= СУММ (B69:D70)

Определите, как изменится по абсолютной величине значение в ячейке **D71** после перемещения содержимого ячейки **C71** в ячейку **C70**.

Задача 8. В электронной таблице значение формулы **=СУММ(B1:B2)** равно 7. Чему равно значение ячейки **B3**, если значение формулы **=СРЗНАЧ(B1:B3)** равно 5?

Задача 9. В электронной таблице значение формулы **=СРЗНАЧ(B1:B2)** равно 2. Чему равно значение ячейки **B3**, если значение формулы **=СУММ(B1:B3)** равно 5?

Задача 10. Дан фрагмент электронной таблицы:

	A	B	C	D
1	3	7	4	= A1+B\$1+ C1
2	9	1	7	

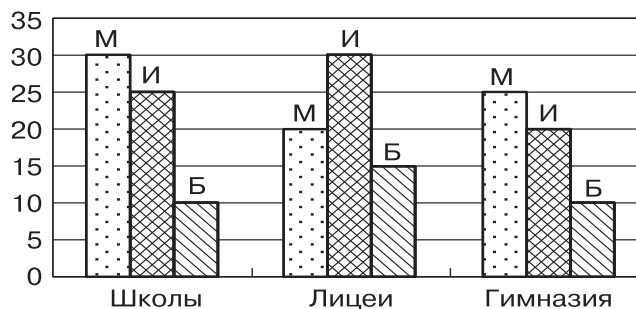
Чему станет равным значение ячейки **D2**, если в нее скопировать формулу из ячейки **C1**? Знак \$ обозначает абсолютную адресацию.

Задача 11. Дан фрагмент электронной таблицы:

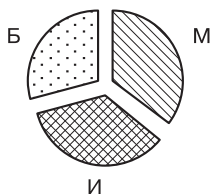
	A	B	C	D
1	5	6	2	= A\$1+B\$1+ 2*C1
2	8	1	6	

Чему станет равным значение ячейки **D2**, если в нее скопировать формулу из ячейки **C1**? Знак \$ обозначает абсолютную адресацию.

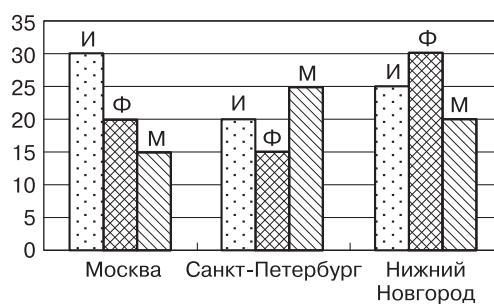
Задача 12. На диаграмме показано количество участников городской олимпиады по трем предметам: математике (М), информатике (И) и биологии (Б):



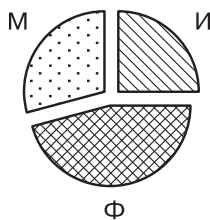
Определите, правильно ли отражает соотношение участников олимпиады по отдельным предметам приведенная ниже диаграмма:



Задача 13. На диаграмме показано количество призеров олимпиады по информатике (И), физике (Ф) и математике (М) из трех городов России:



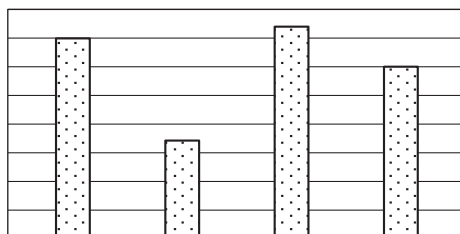
Определите, правильно ли отражает соотношение участников олимпиады по отдельным предметам приведенная ниже диаграмма:



Задача 14. Дан фрагмент электронной таблицы:

	А	В
1	=A4 + B2	5
2	=B1 + 2	2
3	=A1 + 1	7
4	= B1 + B3	

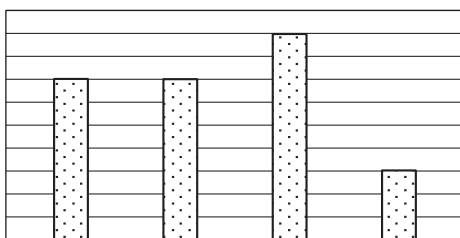
Определите, может ли приведенная ниже диаграмма быть построенной по значениям диапазона ячеек **A1:A4**:



Задача 15. Дан фрагмент электронной таблицы:

	A	B
1	= A3 * 2	6
2	= B2 + 3	2
3	= B1 / 2	
4	= A1 - A3	

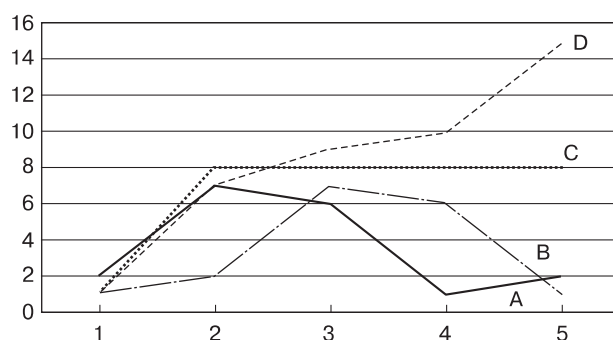
Определите, может ли приведенная ниже диаграмма быть построенной по значениям диапазона ячеек **A1:A4**:



Задача 16. Дан фрагмент электронной таблицы в режиме отображения формул:

	A	B	C	D	E
1	4	6			
2	1	1	1	2	1
3	= A1 + B2	= D2 * \$B\$2	= B1 + E2	= -B2 + \$A\$1*2	= D3 + B2

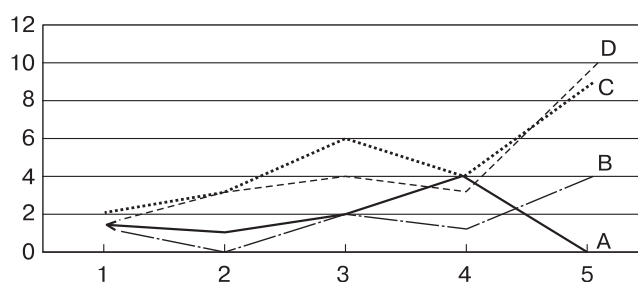
После копирования диапазона ячеек **A3:E3** в диапазон **A4:E6** была построена диаграмма (график) по значениям столбцов диапазона ячеек **B2:E6**. Какой график соответствует значениям ячеек **C2:C6**?



Задача 17. Дан фрагмент электронной таблицы в режиме отображения формул:

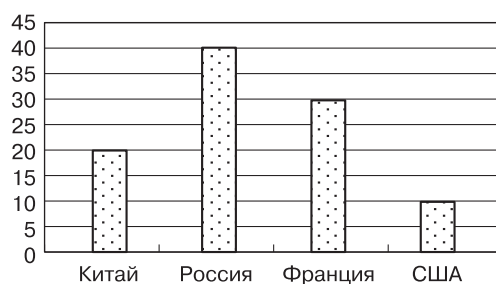
	A	B	C	D	E
1	1	4			
2	1	2	1	2	1
3	=A1 + B2	=C2 * \$A\$3	= -B1 + \$D\$2*2	= B2 + A2	= D2 - E2

После копирования диапазона ячеек **A3:E3** в диапазон **A4:E6** была построена диаграмма (график) по значениям столбцов диапазона ячеек **A2:D6**. Какой график соответствует значениям ячеек **D2:D6**?

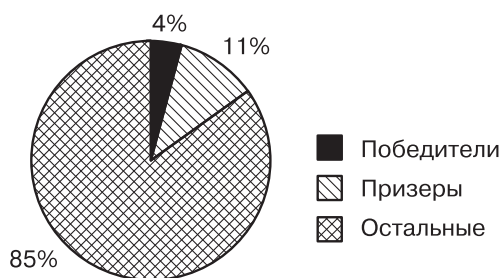


Задача 18. В заочной олимпиаде по математике приняли участие 100 школьников из четырех стран. На диаграммах отражено распределение участников по странам и процентное соотношение победителей и призеров от общего числа участников:

Количество участников из различных стран



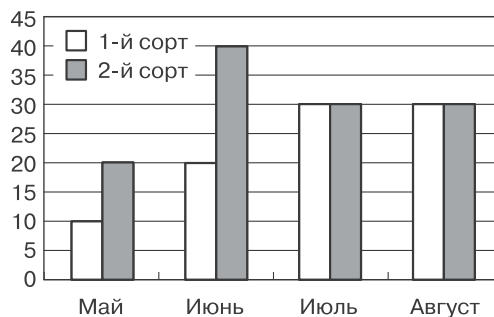
Победители и призеры от общего числа участников



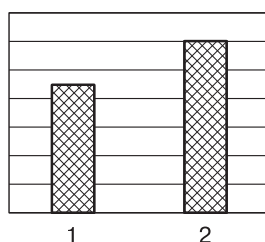
Какое из утверждений следует из приведенных диаграмм?

1. Среди победителей и призеров есть хотя бы 5 человек не из США.
2. Все участники из США стали либо победителями, либо призерами.
3. Хотя бы один школьник из Китая стал призером.
4. Не менее 5 российских школьников стали призерами.

Задача 19. Диаграмма отражает количество (в килограммах) собранного за четыре месяца урожая двух сортов огурцов в парниковом хозяйстве:



Определите, может ли приведенная ниже диаграмма правильно отразить объемы суммарного за четыре месяца собранного урожая по каждому из сортов.



Задача 20. Дан фрагмент электронной таблицы в режиме отображения формул:

	A	B
2	2	=\$A\$1+B\$1*A2
3	1	

Формула, находящаяся в ячейке **B2**, была скопирована в ячейку **B3**. После этого в режиме отображения данных фрагмент электронной таблицы принял вид:

	A	B
2	2	5
3	1	4

Определите значение числа, находящегося в ячейке **A1**.

Задача 21. Дан фрагмент электронной таблицы в режиме отображения формул:

	A	B
2	8	=\$A\$1*\$A2+B\$1
3	9	

Формула, находящаяся в ячейке **B2**, была скопирована в ячейку **B3**. После этого в режиме отображения данных фрагмент электронной таблицы принял вид:

	A	B
2	8	55
3	9	61

Определите значение числа, находящегося в ячейке **A1**.

Задача 22. Дан фрагмент электронной таблицы в режиме отображения формул:

	A	B	C
2	4	=C2*\$A\$1+A2	3
3	5		4

Формула, находящаяся в ячейке **B2**, была скопирована в ячейку **B3**. После этого в режиме отображения данных фрагмент электронной таблицы принял вид:

	A	B	C
2	4	34	3
3	5	45	4

Определите значение числа, находящегося в ячейке **B1**, если известно, что число в ячейке **B1** в 5 раз меньше числа, находящегося в ячейке **A1**.

Организация баз данных

Задача 1. На городской олимпиаде по программированию предлагались задачи трех типов: А, В и С. По итогам олимпиады была составлена таблица, в колонках которой указано, сколько задач каждого типа решил участник. Вот начало этой таблицы:

Фамилия	A	B	C
Иванов	3	2	1

За правильное решение задачи типа А участнику начислялся 1 балл, за решение задачи типа В — 2 балла, а за решение задачи типа С — 3 балла. Победитель определялся по сумме баллов, которая у всех участников оказалась разная. Для определения победителя олимпиады достаточно выполнить следующий запрос:

- 1) отсортировать таблицу по возрастанию значения выражения $A + B + C$ и взять первую строку;
- 2) отсортировать таблицу по возрастанию значения выражения $A + 2B + 3C$ и взять первую строку;
- 3) отсортировать таблицу по убыванию значения выражения $A + 2B + 3C$ и взять первую строку;
- 4) отсортировать таблицу по убыванию значения выражения $A + B + C$ и взять первую строку.

Задача 2. Сколько записей в нижеследующем фрагменте турнирной таблицы удовлетворяют условию: **Место ≤ 3 И (Н > 2 ИЛИ О > 6)**?

Место	Участник	В	Н	П	О
1	Силин	5	3	1	6 $\frac{1}{2}$
2	Клеменс	6	0	3	6
3	Холево	5	1	4	5 $\frac{1}{2}$
4	Яшвили	3	5	1	5 $\frac{1}{2}$
5	Бергер	3	3	3	4 $\frac{1}{2}$
6	Численко	3	2	4	4

Задача 3. Ниже в табличной форме представлен фрагмент базы данных:

№ п/п	Наименование товара	Цена	Количество	Стоимость
1	Монитор	7654	20	153080
2	Клавиатура	1340	26	34840
3	Мышь	235	34	7990
4	Принтер	3770	8	22620
5	Колонки акустические	480	16	7680
6	Сканер планшетный	2880	10	28800

На какой позиции окажется товар «Колонки акустические», если произвести сортировку данной таблицы по возрастанию столбца «Количество»?

Задача 4. Ниже в табличной форме представлен фрагмент базы данных по учащимся 10-х классов:

Фамилия	Имя	Пол	Год рождения	Рост (см)	Вес (кг)
Соколова	Елена	ж	1990	165	51
Антипов	Ярослав	м	1989	170	53
Дмитриева	Елена	ж	1990	161	48
Коровин	Дмитрий	м	1990	178	60
Зубарев	Роман	м	1991	172	58
Полянко	Яна	ж	1989	170	49

Сколько записей в данном фрагменте удовлетворяют условию: **(Имя = "Елена") И (Год рождения > 1989)**?

Задача 5. Ниже в табличной форме представлен фрагмент базы данных:

Номер	Фамилия	Имя	Отчество	Класс	Школа
1	Иванов	Петр	Олегович	10	135
2	Катаев	Сергей	Иванович	9	195
3	Беляев	Иван	Петрович	11	45
4	Носов	Антон	Павлович	7	4

Какую строку будет занимать ученик с фамилией «Иванов» после проведения сортировки по убыванию по полю «Класс»?

Задача 6. Ниже в табличной форме представлен фрагмент базы данных:

	Название пролива	Длина (км)	Ширина (км)	Глубина (м)	Местоположение
1	Босфор	30	0,7	20	Атлантический океан
2	Магелланов	575	2,2	29	Тихий океан
3	Ормузский	195	54	27	Индийский океан
4	Гудзонов	806	115	141	Северный Ледовитый океан
5	Гибралтарский	59	14	53	Атлантический океан
6	Ла-Манш	578	32	23	Атлантический океан
7	Баб-эль-Мандебский	109	26	31	Индийский океан
8	Дарданеллы	120	1,3	29	Атлантический океан
9	Берингов	96	86	36	Тихий океан

Сколько записей в данном фрагменте удовлетворяют условию: **(Длина (км) > 100 ИЛИ Глубина (м) > 25) И (Местоположение = Атлантический океан)?**

Задача 7. Представлена часть базы данных «Расписание уроков»:

№	День	№ урока	Кабинет	Предмет	Учитель	Класс
1	пн	1	22	геомет	Голубева	5б
2	пн	2	21	географ	Иванова	4в
3	вт	4	23	информ	Зайцев	11б
4	вт	3	24	физика	Зайцев	8а
5	чт	4	35	физика	Калугина	10б
6	пт	3	32	матем	Петров	8а

Определите, какие записи удовлетворяют условию отбора **Класс > "11а"**.

Задача 8. База данных школы включает таблицу с информацией о золотых и серебряных медалистах, которая, наряду с другими, имеет поля с названиями «Год» и «Тип медали». В таблице находятся записи о количестве медалистов в 2008, 2009 и 2010 гг. Всего таблица включает 36 записей.

Заведующий учебной частью запросил информацию о медалистах. Ниже приведены запросы и количество записей таблицы (N), которые удовлетворяют этим запросам:

Запрос	N
Год = 2009 ИЛИ Тип медали = золотая	25
Неверно, что (Тип медали = золотая И Год = 2008)	32
Тип медали = серебряная И Год = 2010	8

Определите количество записей, удовлетворяющих запросу **Тип медали = золотая И Год = 2010**.

Задача 9. База данных школы включает таблицу с информацией о золотых и серебряных медалистах, которая, наряду с другими, имеет поля с названиями «Год» и «Тип медали». В таблице находятся записи о количестве медалистов в 2008, 2009 и 2010 гг.

Заведующий учебной частью запросил информацию о медалистах. Ниже приведены запросы и количество записей таблицы (N), которые удовлетворяют этим запросам.

Запрос	N
Тип медали = серебряная	18
Неверно, что (Тип медали = золотая И Год = 2010)	31
Тип медали = золотая ИЛИ Год \neq 2009	29
Год = 2008 ИЛИ Тип медали = серебряная	3
Год = 2010 ИЛИ Тип медали = серебряная	8

Определите количество записей, удовлетворяющих запросу **Год \neq 2008**.

Задача 10. Таблица «Чайники», наряду с другими, включает поля с названиями «Тип чайника» и «Объем». В базе данных находятся 31 запись об электрических чайниках и чайниках для газовой плиты. Количества записей N , удовлетворяющих различным запросам, приведены ниже:

Запрос	<i>N</i>
Тип чайника = для газовой плиты И Объем \neq 2	6
Неверно, что (Тип чайника = для газовой плиты ИЛИ Объем = 2)	13
Тип чайника = электрический И Объем = 2	7

Определите количество записей, удовлетворяющих запросу **Тип чайника = для газовой плиты**.

Задача 11. Таблица «Чайники», наряду с другими, включает поля с названиями «Тип чайника» и «Объем». В базе данных находятся записи об электрических чайниках и чайниках для газовой плиты. Количество записей *N*, удовлетворяющих различным запросам, приведены ниже:

Запрос	<i>N</i>
Тип чайника = электрический И Объем $>$ 1	10
Тип чайника = для газовой плиты И Объем = 2	5
Неверно, что (Тип чайника = для газовой плиты И Объем = 3)	29

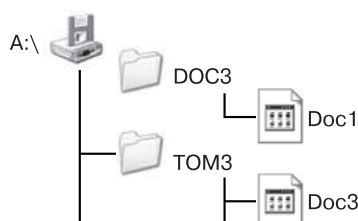
Определите количество записей, удовлетворяющих запросу **Объем = 1**.

Организация файловых систем

Задача 1. Определите, удовлетворяет ли имя файла **datt2** маске **?a*****.

Задача 2. В некотором каталоге хранился файл **Задание3**. После того как в этом каталоге создали подкаталог и переместили в созданный подкаталог файл **Задание3**, полное имя файла стало **Е:\Школа\Физика\Задания\Задание3**. Каково было полное имя этого файла до перемещения?

Задача 3. Дано дерево каталогов. Определите полное имя файла **Doc3**.



Задача 4. Пользователь, перемещаясь из одного каталога в другой, последовательно посетил каталоги **TASKS**, **DOCUMENTS**, **SCHOOL**, **D:**, **MYDOC**, **LETTERS**. При каждом перемещении пользователь либо спускался в каталог на один уровень ниже, либо поднимался на один уровень выше. Каково полное имя каталога, из которого начал перемещение пользователь?

Задача 5. Каталог содержит файлы с именами

а) **p5.pas**; б) **p4.ppt**; в) **pq.pas**; г) **pq.p**; д) **pr.p**; е) **p12.ppt**.

Определите, в каком порядке будут показаны эти файлы, если выбрана сортировка по типу (по возрастанию).

Задача 6. Определите, удовлетворяет ли имя файла **ddbaddg** маске **?d??g***.

Глава 4

Алгоритм. Исполнитель алгоритма

Алгоритм представляет собой сформулированную на некотором языке последовательность действий, выполнение которой позволяет получить на основе исходных данных искомый результат. С понятием алгоритма связано понятие «*исполнитель алгоритма*» — некоторый субъект (человек или устройство; в частности, в качестве исполнителя может выступать процессор ЭВМ), способный выполнять определенный набор действий.

Исполнителя характеризуют: среда, система команд и отказы.

Среда содержит объекты, с которыми исполнитель взаимодействует в процессе выполнения заданного алгоритма (например, процессор взаимодействует с оперативной памятью и разнообразными внешними устройствами).

Система команд — набор действий, которые может выполнять исполнитель. Например, в систему команд большинства процессоров входят операции сложения целых и вещественных чисел. После вызова команды исполнитель совершает соответствующее действие. Например, после вызова команды сложения процессор выполняет суммирование двух чисел.

Отказы исполнителя возникают, если некоторая команда вызывается при недопустимом для нее состоянии среды. Например, отказ процессора возникает при попытке деления некоторого числа на нуль. Другой причиной отказа может быть несоответствие заданной команды системе команд исполнителя.

Наиболее распространенными являются следующие формы *записи алгоритмов*:

- 1) графическая запись (блок-схема);
- 2) словесная запись (псевдокод);
- 3) язык программирования.

Словесная форма записи алгоритма (псевдокод) представляет собой описание последовательности этапов обработки данных на языке, приближенном к естественному.

Графическая форма записи, называемая также *блок-схемой алгоритма*, представляет собой изображение алгоритма в виде после-

довательности связанных между собой функциональных блоков, каждый из которых соответствует определенному действию. Эти блоки соединяются линиями переходов, определяющими очередность выполнения действий.

Язык программирования — формальная знаковая система, предназначенная для записи компьютерных программ. Язык программирования позволяет определить набор действий, которые должен выполнить исполнитель алгоритма.

Компьютерная программа — набор инструкций, записанных с помощью определенного алфавита и предназначенных для исполнения устройством управления вычислительной машины. Программа может быть записана в машинных кодах, на языке низкого уровня или языке высокого уровня.

При использовании *машинных кодов* текст программы представляет собой двоичные коды (набор нулевых и единичных битов), каждый из которых соответствует определенной операции. В настоящее время запись программ в машинных кодах не практикуется, так как процесс создания программы в этом случае очень трудоемок и не нагляден.

Язык программирования низкого уровня (ассемблер) близок к машинным кодам. Версия языка ассемблера определяется процессором, для программирования которого он предназначен. Использовать ассемблер можно лишь при наличии знаний об архитектуре процессора. Отличие же ассемблера от машинных кодов заключается в том, что двоичные коды операций заменяются словесными обозначениями (*мнемокодом*), обозначающими суть операции, а двоичные адреса операндов заменяются именами переменных.

Язык программирования высокого уровня разработан для описания решения задач в наглядном виде. Основная особенность любого языка высокого уровня — использование конструкций, описывающих различные структуры данных и операции над ними. (Обычно описания таких структур в машинных кодах или на языке программирования низкого уровня слишком длинны и сложны для понимания.) Примерами языков высокого уровня являются C, C++, Visual Basic, Java, Python, PHP, Ruby, Perl, Pascal.

Необходимо отметить, что на каком бы языке программирования ни была написана программа, перед выполнением она всегда переводится в машинные коды.

Задача 1. Строки (цепочки символов латинских букв) создаются по следующему правилу. Первая строка состоит из одного символа — латинской буквы «А». Каждая из последующих цепочек создается такими действиями: в очередную строку сначала записывается буква, чей порядковый номер в алфавите соответствует номеру строки (на i -м шаге пишется i -я буква алфавита), а затем к ней справа дважды подряд приписывается предыдущая строка. Вот первые 4 строки, созданные по этому правилу:

(1): А

(2): БАА

(3): СВААВАА

(4): DCBAABAACBAABAА

Латинский алфавит (для справки):

ABCDEFGHIJKLMNOPQRSTUVWXYZ.

Запишите семь символов подряд, стоящие в восьмой строке с 126-го по 132-е место (считая слева направо) [6].

Решение.

Найдем количество символов в нескольких первых строках:

(1): А — 1 символ;

(2): БАА — 3 символа;

(3): СВААВАА — 7 символов;

(4): DCBAABAACBAABAА — 15 символов.

Отметим, что каждая строка, начиная со второй, состоит из трех частей:

1) i -я буква алфавита (i — номер строки), записываемая в начало строки;

2) предыдущая строка;

3) предыдущая строка.

Количество символов в i -й строке вычисляется по формуле: $N_i = 2^i - 1$. Следовательно, в восьмой строке содержится $N_8 = 2^8 - 1 = 255$ символов.

В табл. 4.1 указано количество символов в i -й строке (для каждой ее части), а в скобках указаны номера символов в строке.

Таблица 4.1

№ строки/ кол. симв.	1-я часть	2-я часть	3-я часть	Всего символов
2	1 (1)	1 (2)	1 (3)	3
3	1 (1)	3 (2–4)	3 (5–7)	7
4	1 (1)	7 (2–8)	7 (9–15)	15
5	1 (1)	15 (2–16)	15 (17–31)	31
6	1 (1)	31 (2–32)	31 (33–63)	63
7	1 (1)	63 (2–64)	63 (65–127)	127
8	1 (1)	127 (2–128)	127 (129–255)	255

Из табл. 4.1 видно, что восьмая строка состоит из символа латинского алфавита под номером 8 (т. е. символа «Н») и двух повторений седьмой строки. Учитывая это, можно отметить следующее:

1) последние три символа с номерами 126, 127 и 128 в восьмой строке соответствуют последним трем символам седьмой строки (с номерами 125–127);

2) 129-й, 130-й, 131-й и 132-й символы восьмой строки соответствуют первым четырем символам седьмой строки (1–4).

Найдем первые четыре символа седьмой строки. В начало каждой строки записывается символ латинского алфавита, номер которого соответствует номеру строки. Следовательно, в начале седьмой строки находятся символы алфавита с 7-го по 1-й: GFEDCBA. Таким образом, первые четыре символа седьмой строки, и, соответственно, символы в восьмой строке с 129 по 132: GFED.

Найдем последние три символа в седьмой строке. Выполнив анализ второй, третьей и четвертой строк, можно отметить, что они заканчиваются на ВАА. Учитывая, что при составлении новой строки модифицируется только ее начало, а окончание остается неизменным, седьмая строка также заканчивается на ВАА. Следовательно, 126-й, 127-й и 128-й символы восьмой строки — ВАА.

Таким образом, в восьмой строке символы со 126-го по 132-й это символы BAAGFED.

Ответ: BAAGFED.

Задача 2. Цепочки символов (строки) создаются по следующему правилу. Первая строка состоит из одного символа — цифры «1». Каждая из последующих цепочек создается следующим действием: в очередную строку дважды записывается предыдущая цепочка цифр (одна за другой, подряд), а в конец приписывается еще одно число — номер строки по порядку (на i -м шаге дописывается число i). Вот первые 4 строки, созданные по этому правилу:

(1): 1

(2): 112

(3): 1121123

(4): 112112311211234

Сколько раз в общей сложности встречаются в восьмой строке четные цифры (2, 4, 6, 8) [4]?

Решение.

Найдем количество четных цифр в первых строках:

(1): 1 — 0 цифр;

(2): 112 — 1 цифра;

(3): 1121123 — 2 цифры ($2 = 1 \cdot 2$);

(4): 112112311211234 — 5 цифр ($5 = 2 \cdot 2 + 1$).

Допустим, в i -й строке содержится K_i цифр. Запишем формулы для вычисления количества символов в строке:

1) с нечетным номером: $K_i = 2 \cdot K_{i-1}$;

2) с четным номером: $K_i = 2 \cdot K_{i-1} + 1$.

Вычислим количество четных цифр в 5-й — 8-й строках:

(5): 10 цифр ($10 = 5 \cdot 2$);

(6): 21 цифра ($21 = 10 \cdot 2 + 1$);

(7): 42 цифры ($42 = 21 \cdot 2$);

(8): 85 цифр ($85 = 42 \cdot 2 + 1$).

Ответ: 85.

Задача 3. Цепочки символов (строки) создаются по следующему правилу. Первая строка состоит из одного символа — цифры «1». Каждая из последующих цепочек создается такими действиями: в очередную строку дважды записывается цепочка цифр из предыдущей строки (одна за другой, подряд), а в конец приписывается еще одно число — номер строки по порядку (на i -м шаге дописывается число i). Вот первые 4 строки, созданные по этому правилу:

(1): 1

(2): 112

(3): 1121123

(4): 112112311211234

Какая цифра стоит в седьмой строке на 120-м месте (считая слева направо) [3]?

Решение.

Определим количество цифр в каждой строке:

(1): 1 — 1 цифра;

(2): 112 — 3 цифры;

(3): 1121123 — 7 цифр;

(4): 112112311211234 — 15 цифр.

Количество цифр в i -й строке вычисляется по формуле: $N_i = 2^i - 1$. Получаем, что в седьмой строке содержится $N_7 = 2^7 - 1 = 127$ цифр.

В конец каждой строки записывается цифра, соответствующая номеру строки, т.е. седьмая строка заканчивается на «...11211234567». Тогда можно записать:

№ цифры в строке	120	121	122	123	124	125	126	127
Значение цифры	1	1	2	3	4	5	6	7

Следовательно, на 120 месте стоит цифра «1».

Ответ: 1.

Задача 4. Система команд исполнителя РОБОТ, «живущего» в прямоугольном лабиринте на клетчатой плоскости: *вверх*, *вниз*, *влево*, *вправо*. При выполнении любой из этих команд РОБОТ перемещается на одну клетку соответственно: *вверх* \uparrow , *вниз* \downarrow , *влево* \leftarrow , *вправо* \rightarrow .

Четыре команды проверяют истинность условия отсутствия стены у каждой стороны той клетки, где находится РОБОТ: *сверху свободно*, *снизу свободно*, *слева свободно*, *справа свободно*.

Цикл «**ПОКА** < *условие* > *команда* » выполняется, пока *условие* истинно, иначе происходит переход на следующую строку.

Сколько клеток лабиринта соответствуют требованию, что, выполнив предложенную ниже программу, РОБОТ остановится в той же клетке, с которой он начал движение [6]?

НАЧАЛО

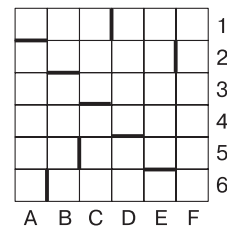
ПОКА < *снизу свободно* > *вниз*

ПОКА < *слева свободно* > *влево*

ПОКА < *сверху свободно* > *вверх*

ПОКА < *справа свободно* > *вправо*

КОНЕЦ



Решение.

Последнее действие программы указывает на клетки, которые являются возможными решениями задачи. Это позволяет существенно сократить время поиска, исключив перебор всех вариантов решения (всех клеток лабиринта).

Последнее действие РОБОТА в программе — **ПОКА** < *справа свободно* > *вправо*. Следовательно, РОБОТ завершит движение, встретив стенку справа. В качестве возможных решений отметим и рассмотрим подробнее те клетки лабиринта, которые ограничены стенкой справа (рис. 4.1, а клетки помечены точками).

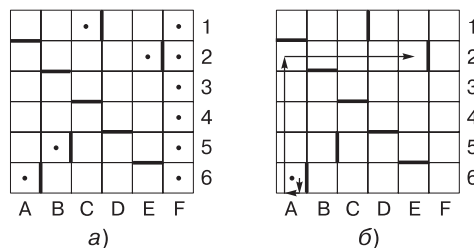


Рис. 4.1. Анализ клеток лабиринта: а) возможные решения задачи; б) выполнение программы для клетки **A1**

Допустим, РОБОТ начал движение из клетки **A1**, тогда его путь завершится в клетке **E5**. Так как эти клетки не совпадают, клетка **A1** не является решением задачи (см. рис. 4.1, б).

Проверив таким же способом оставшиеся клетки, мы получим, что только клетка **F3** является решением задачи (рис. 4.2).

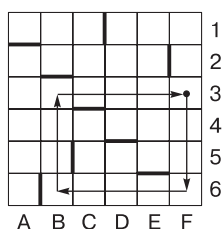


Рис. 4.2. Выполнение программы для клетки **F4**

Ответ: 1.

Задача 5. Система команд исполнителя РОБОТ, «живущего» в прямоугольном лабиринте на клетчатой плоскости: *вверх*, *вниз*, *влево*, *вправо*. При выполнении этих команд РОБОТ перемещается на одну клетку соответственно: *вверх* ↑, *вниз* ↓, *влево* ←, *вправо* →.

Четыре команды проверяют истинность условия отсутствия стены у той клетки, где находится РОБОТ: *сверху свободно*, *снизу свободно*, *слева свободно*, *справа свободно*.

Цикл **ПОКА** < *условие* > *команда* выполняется, пока *условие* истинно, иначе происходит переход на следующую строку.

Если РОБОТ начнет движение в сторону стены, то он разрушится и программа прервется. Сколько клеток лабиринта соответствуют требованию, что, выполнив предложенную программу, РОБОТ уцелеет и остановится в той же клетке, с которой он начал движение [7]?

НАЧАЛО

ПОКА < *сверху свободно* > *вправо*

ПОКА < *справа свободно* > *вниз*

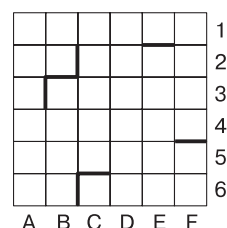
ПОКА < *снизу свободно* > *влево*

ПОКА < *слева свободно* > *вверх*

КОНЕЦ

Решение.

Последняя команда в программе РОБОТА — **ПОКА** < *слева свободно* > *вверх*. Следовательно, РОБОТ завершит программу в той клетке, у которой есть стенка слева (рис. 4.3, а, клетки отмечены точками).



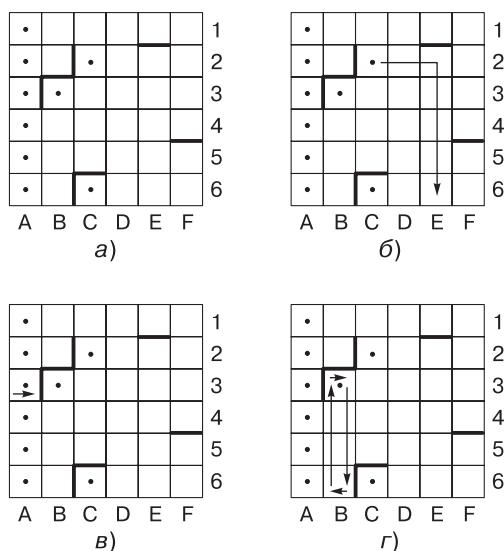


Рис. 4.3. Анализ клеток лабиринта

Рассмотрим процесс выполнения программы для клетки **С2** (см. рис. 4.3, б):

1) команда **ПОКА** *< сверху свободно > вправо* — сверху этой клетки нет стенки, поэтому РОБОТ начнет движение и переместится до клетки **Е2**, у которой сверху стенка есть;

2) команда **ПОКА** *< справа свободно > вниз* — справа клетки **Е2** нет стенки, поэтому РОБОТ начнет движение вниз. Достигнув клетки **Е6**, программа заикнется, так как справа от этой клетки нет стенки, но и продвинуться за границы лабиринта РОБОТ не может. Поэтому клетка **С2** не является решением задачи.

Рассмотрим клетку **А3**. Вверху данной клетки нет стенки, поэтому РОБОТ перейдет к выполнению первой команды программы: начнет движение в сторону стенки и разрушится (рис. 4.3, в). Поэтому клетка **А3** не является решением задачи.

Рассмотрим теперь процесс выполнения программы для клетки **В3** (см. рис. 4.3, г):

1) команда **ПОКА** *< сверху свободно > вправо* — сверху данной клетки есть стенка, поэтому РОБОТ сразу перейдет к выполнению следующей команды программы (РОБОТ не начинает движение в сторону стенки, поэтому и не разрушается, — РОБОТ только проверяет наличие стенки сверху);

2) команда **ПОКА** *< справа свободно > вниз* — справа от клетки **В3** нет стенки, поэтому РОБОТ начнет движение вниз. Достигнув

клетки **В6**, РОБОТ остановится (справа от данной клетки есть стенка), после чего перейдет к выполнению следующей команды программы;

3) команда **ПОКА** *< снизу свободно > влево* — внизу клетки **В6** есть стенка (граница лабиринта), поэтому РОБОТ сразу переходит к выполнению следующей команды;

4) команда **ПОКА** *< слева свободно > вверх* — слева от клетки **В6** нет стенки, поэтому РОБОТ начнет движение вверх. Достигнув клетки **В3**, РОБОТ остановится, так как слева от данной клетки есть стенка. Таким образом, клетка **В3** является решением задачи, так как РОБОТ начинает и завершает выполнение программы в данной клетке.

Анализируя аналогичным способом оставшиеся клетки, мы получим, что, за исключением клетки **В3**, нет клеток, удовлетворяющих условию задачи.

Ответ: 1.

Задача 6. Исполнитель **ЧЕРЕПАШКА** перемещается на экране компьютера, оставляя след в виде линии. В каждый конкретный момент известно положение исполнителя и направление его движения. У исполнителя существуют следующие команды:

Вперед n , где n — целое число, вызывающее передвижение черепашки на n шагов в направлении движения;

Направо m , где m — целое число, вызывающее изменение направления движения на m градусов по часовой стрелке.

Запись **Повтори 5 [Команда1 Команда2]** означает, что последовательность команд, записываемых в скобках, повторится 5 раз. Исполнитель интерпретирует эту запись как одну команду.

ЧЕРЕПАШКЕ был дан для исполнения следующий алгоритм:

Повтори 2 [Повтори 4[Вперед 40 Направо 90] Направо 180].

Что появится на экране после выполнения указанного алгоритма?
Решение.

ЧЕРЕПАШКА после выполнения четырех команд **[Вперед 40 Направо 90]** поворачивается на 360° ($90 + 90 + 90 + 90 = 360$) и возвращается в исходную точку. В данном случае **ЧЕРЕПАШКА** рисует четыре линии, которые находятся по отношению друг к другу под углом 90° , т. е. рисует квадрат.

Затем **ЧЕРЕПАШКА** меняет направление движения, выполняя поворот на 180° . Всего, как следует из команды, **ЧЕРЕПАШКА** меняет направление движения 2 раза. Учитывая, что $180 = 360 / 2$, **ЧЕРЕПАШКА** нарисует два квадрата.

Ответ: два квадрата.

Задача 7. Имеется исполнитель КУЗНЕЧИК, который «живет» на числовой оси. Система команд КУЗНЕЧИКА: **Вперед N** (КУЗНЕЧИК прыгает вперед на N единиц); **Назад M** (КУЗНЕЧИК прыгает назад на M единиц). Переменные N и M могут принимать любые целые положительные значения. Известно, что КУЗНЕЧИК выполнил программу из 50 команд, в которой команд Назад 2 на 12 больше, чем команд Вперед 3. Других команд в программе не было. На какую одну-единственную команду можно заменить эту программу, чтобы КУЗНЕЧИК оказался в той же точке, что и после выполнения программы [1]?

Решение.

Выполнив программу, КУЗНЕЧИК переместился от некоторой начальной точки вперед на $19 \cdot 3 = 57$ единиц, а назад — на $31 \cdot 2 = 62$ единицы. Значит, КУЗНЕЧИК от начального положения переместился назад на $62 - 57 = 5$ единиц. Следовательно, исходную программу можно заменить командой **Назад 5**.

Ответ: **Назад 5**.

Задача 8. Исполнитель ЧЕРЕПАШКА перемещается на экране компьютера, оставляя след в виде линии. В каждый конкретный момент известно положение исполнителя и направление его движения. У исполнителя существует две команды:

1) **Вперед n** , вызывающая передвижение ЧЕРЕПАШКИ на n шагов в направлении движения;

2) **Направо t** , вызывающая изменение направления движения на t градусов по часовой стрелке ($0 \leq t \leq 180$).

Вместо n и t должны стоять целые числа.

Запись: **Повтори 5 [Команда1 Команда2]** означает, что последовательность команд в квадратных скобках повторится 5 раз.

Какое число необходимо записать вместо n в следующем алгоритме: **Повтори 7 [Вперед 40 Направо n]**, чтобы на экране появился правильный шестиугольник [3]?

Решение.

Команда **[Вперед 40 Направо n]** рисует линию заданной длины и задает направление движения по часовой стрелке. Но именно направление движения определяет величину внешнего угла отображаемого многоугольника. Внешний угол правильного многоугольника вычисляется по формуле: $\alpha = 360 / n$, где n — число сторон многоугольника (рис. 4.4). Таким образом, чтобы на экране появился правильный шестиугольник, в команде **Направо n** значение n должно быть равно 60° .

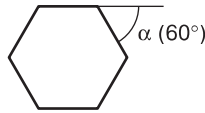


Рис. 4.4. Результат построения фигуры

Всего же для построения правильного шестиугольника достаточно 6 команд **[Вперед 40 Направо n]**, а седьмая команда в заданном алгоритме рисует прямую, которая повторяет первую.

Ответ: 60.

Задача 9. В приведенном ниже фрагменте алгоритма, записанном на алгоритмическом языке, тип переменных a, b, c — «строка», а переменных i, k — «целое». Используются следующие функции:

1) **Длина(a)** — возвращает количество символов в строке a (тип «целое»);

2) **Извлечь(a, i)** — возвращает i -й (слева) символ в строке a (тип «строка»);

3) **Склеить(a, b)** — возвращает строку, в которой записаны сначала все символы строки a , а затем все символы строки b (тип «строка»).

Значения строк записываются в одинарных кавычках (например, $a := \text{'дом'}$). Фрагмент алгоритма:

```

i := 1
b := 'П'
пока i <= Длина(a)
нц
  c := Извлечь(a, i)
  b := Склеить(b, c)
  i := i + 1
кц

```

Какое значение будет у переменной b после выполнения приведенного выше фрагмента алгоритма, если значение переменной a было 'РОЗА'?

Решение.

Значения переменных a и k , как видно из текста программы, не изменяются в ходе ее выполнения.

До начала цикла вычисляются значения переменных i, k и b :

- 1) $i := \text{Длина}(a)$, $i = 4$;
- 2) $k := 1$, $k = 1$;
- 3) $b := \text{'П'}$, $b = \text{'П'}$.

Рассмотрим процесс выполнения цикла по шагам:

Шаг	Оператор	i	b	c
1	$c := \text{Извлечь}(a, i)$	1	'П'	'Р'
	$b := \text{Склеить}(b, c)$	1	'ПР'	'Р'
	$i := i + 1$	2	'ПР'	'Р'
2	$c := \text{Извлечь}(a, i)$	2	'ПР'	'О'
	$b := \text{Склеить}(b, c)$	2	'ПРО'	'О'
	$i := i + 1$	3	'ПРО'	'О'
3	$c := \text{Извлечь}(a, i)$	3	'ПРО'	'З'
	$b := \text{Склеить}(b, c)$	3	'ПРОЗ'	'З'
	$i := i + 1$	4	'ПРОЗ'	'З'
4	$c := \text{Извлечь}(a, i)$	4	'ПРОЗ'	'А'
	$b := \text{Склеить}(b, c)$	4	'ПРОЗА'	'А'
	$i := i + 1$	5	'ПРОЗА'	'А'

Таким образом, в результате выполнения приведенного фрагмента программы значение переменной b будет равно 'ПРОЗА'.

Ответ: 'ПРОЗА'.

Задача 10. Исполнитель ПРОЦЕССОР имеет два регистра с именами A и B , в которых хранятся вещественные числа. В систему команд ПРОЦЕССОРА входят 6 команд:

?An	Ввод числа n в ячейку A
?Bn	Ввод числа n в ячейку B
!A	Вывод данных из ячейки A на экран
!B	Вывод данных из ячейки B на экран
*BA	Перемножить содержимое ячеек B и A и полученный результат поместить в ячейку B . Содержимое ячейки A остается неизменным
*AB	Перемножить содержимое ячеек B и A и полученный результат поместить в ячейку A . Содержимое ячейки B остается неизменным
/BA	Разделить содержимое ячейки B на содержимое ячейки A и полученный результат поместить в ячейку B . Содержимое ячейки A остается неизменным
/AB	Разделить содержимое ячейки A на содержимое ячейки B и полученный результат поместить в ячейку A . Содержимое ячейки B остается неизменным

Какие числа будут выведены на экран после выполнения команд:
?A4 ?B16 /BA *AB /BA !A !B ?

Решение.

Выполним указанную последовательность команд:

Шаг	Команда	Значение переменной	
		A	B
1	?A4 ($A = 4$)	4	?
2	?B16 ($B = 16$)	4	16
3	/BA ($B = 16 / 4 = 4$)	4	4
4	*AB ($A = 4 \cdot 4 = 16$)	16	4
5	/BA ($B = 4 / 16 = 0,25$)	16	0,25

После выполнения команд на экран будет выведено сначала значение переменной A , а затем — переменной B : 16 0,25.

Ответ: 16 0,25.

Задача 11. Исполнитель КАЛЬКУЛЯТОР, разработанный школьниками на уроках программирования, имеет три регистра (их имена: A , B , C) и позволяет выполнять с регистрами поразрядные операции. Система команд исполнителя:

- 1) $>A$ — ввод данных в регистр;
- 2) $<A$ — вывод данных из регистра;
- 3) $A + B$ — сохранить без изменения нулевые разряды регистра B , соответствующие единичным разрядам регистра A , а остальные разряды регистра B инвертировать.

Определите функцию, которая вычисляется следующей программой:

$>A >B C + B B + B B + C A + A C + A <A$.

Решение.

В заданной программе в первой команде используется регистр C , значения которого не вводятся, следовательно, регистр C изначально заполнен нулями.

Построим таблицу, в которую будем записывать результаты выполнения команд программы:

$>A$	$>B$	C	$C+B$	$B+B$	$B+C$	$A+A$	$C+A$
0	0	0	1	0	1	1	0
1	0	0	1	0	1	0	0
0	1	0	0	1	0	1	0
1	1	0	0	1	0	0	1
Регистр, в который записывается результат			B	B	C	A	A

С помощью СДНФ построим по последнему столбцу логическое выражение. Получим: $F = A \wedge B$.

Ответ: $F = A \wedge B$.

Задача 12. Исполнитель РОБОТ действует на клетчатой доске, между соседними клетками которой могут стоять стены. РОБОТ передвигается по клеткам доски и может выполнять команды: 1 — вверх, 2 — вниз, 3 — вправо, 4 — влево, переходя на соседнюю клетку в указанном направлении. Если в этом направлении между клетками стоит стена, то РОБОТ разрушается.

РОБОТ успешно выполнил программу **33233241**. Какую последовательность из четырех команд должен теперь выполнить РОБОТ, чтобы вернуться в ту же самую клетку, с которой он начал выполнение программы, и не разрушиться [3]?

Решение.

Изобразим путь РОБОТА (рис. 4.5) и запишем этот путь от конечной точки до начальной. В условии задачи сказано, что на клетчатой доске существуют стенки. Успешно выполненная программа говорит о том, что на пути, по которому прошел робот, стенки отсутствуют. Поэтому при записи пути РОБОТА из конечной клетки в начальную можно использовать только те клетки, по которым РОБОТ двигался ранее.

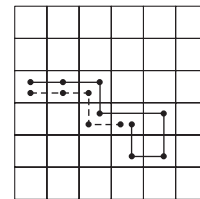


Рис. 4.5. Путь РОБОТА

На рис. 4.5 сплошной линией показан путь РОБОТА, который он прошел в соответствии с заданной программой, а пунктирной — путь, который он пройдет, возвращаясь в исходную точку. Для этого РОБОТ должен выполнить команды **4144**.

Ответ: 4144.

Задача 13. У исполнителя КАЛЬКУЛЯТОР имеются две команды, которым присвоены номера:

1: прибавь 3;

2: умножь на 4.

Выполняя первую из них, КАЛЬКУЛЯТОР прибавляет к числу на экране 3, а выполняя вторую, умножает его на 4. Запишите порядок команд в программе получения из числа 3 числа 57, содержащей не более 6 команд, указывая только номера этих команд. (Например, программа **21211** — это программа: умножь на 4, прибавь 3, умножь на 4, прибавь 3, прибавь 3, которая преобразует число 2 в 50) [6].

Решение.

В этой задаче необходимо из числа 3 получить число 57, используя две операции.

Существует два способа решения задачи: прямой (определение команд для получения из числа 3 числа 57) и обратный (вычисление команд для получения из 57 числа 3). Рассмотрим обратный способ решения (рис. 4.6).

Запишем действия, выполняемые при решении задачи:

1) найдем число x , из которого получено число 57. Его можно получить из x , прибавив к нему 3 ($57 = x + 3$) или умножив x на 4 ($57 = x \cdot 4$). Уравнение $x \cdot 4 = 57$ в целых числах решения не имеет, т. е. не существует целого числа, умножив которое на 4, можно получить 57. Уравнение же $x + 3 = 57$ имеет решение: $x = 54$. Следовательно, число 57 получено из числа 54, прибавлением к нему 3 (команда 1);

2) рассмотрим число 54. Запишем уравнения: $54 = x \cdot 4$ и $54 = x + 3$. Первое уравнение в целых числах решения не имеет. Из второго уравнения получим, что $x = 51$. Следовательно, 54 получено из 51 прибавлением к нему 3, т. е. с использованием команды с номером 1;

3) рассмотрим число 51: $51 = x \cdot 4$ или $51 = x + 3$. Число 51 не делится на 4, но его можно получить из 48, используя первую команду: $51 = 48 + 3$ (команда 1);

4) рассмотрим число 48: $x + 3 = 48 \Rightarrow x = 45$ или $x \cdot 4 = 48 \Rightarrow x = 12$. Оба уравнения имеют решения в целых числах. Следовательно, число 48 можно получить из 45 (команда 1) или из 12 (команда 2, см. рис. 4.6).

Указанная возможность получения 48 с помощью двух различных операций отображается на дереве в виде ветвления. Далее мы будем анализировать числа 12 и 45.

5.1) рассмотрим число 45: $45 = x_1 + 3 \Rightarrow x_1 = 37$ и $45 = x_2 \cdot 4$. Только первое из двух этих уравнений имеет решение в целых числах, поэтому число 45 было получено из числа 42 (команда 1);

5.2) рассмотрим число 12: $12 = x_1 + 3 \Rightarrow x_1 = 9$ и $12 = x_2 \cdot 4 \Rightarrow x_2 = 3$. Таким образом, число 12 может быть получено из исходного числа 3 (команда 2).

Таким образом, для получения числа 57 из 3 использованы команды: **22111** (порядок записи команд — обратный).

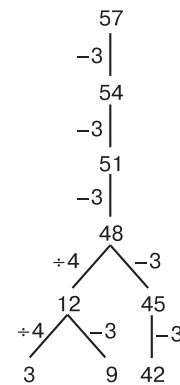


Рис. 4.6. Дерево анализа операций

Отметим, что эту задачу можно решить и без записи уравнений, производя указанные действия в уме.

Ответ: 22111.

Задача 14. У исполнителя ПРОЦЕССОР есть две ячейки памяти, в каждой из которых записано по числу, и две команды, которым присвоены номера:

1 — запиши сумму чисел в первую ячейку;

2 — запиши сумму чисел во вторую ячейку.

Выполняя первую команду, ПРОЦЕССОР складывает числа в ячейках и заменяет этой суммой число в первой ячейке, а выполняя вторую — складывает числа и заменяет этой суммой число во второй ячейке. Запишите порядок команд в программе получения из пары чисел 1 и 2 пары чисел 19 и 8, содержащей не более 5 команд, указывая только номера этих команд.

Решение.

Пусть запись (a, b) обозначает, что в первой ячейке хранится число a , а во второй — число b .

Существует два способа решения задачи: прямой — получение из пары (1,2) пары (19,8) и обратный — получение из пары (19,8) пары (1,2).

1-й способ (прямой).

В паре (19,8) с помощью указанных команд нужно сначала получить 8, так как $8 < 19$; иначе, если сначала получить 19, число 8 получить будет невозможно:

1) сложим 1 и 2, запишем сумму в первую ячейку, получим пару (3,2) — команда 1;

2) сложим 3 и 2, запишем сумму во вторую ячейку: (3,5) — команда 2;

3) сложим 3 и 5, запишем сумму во вторую ячейку: (3,8) — команда 2; Получим 19 в первой ячейке;

4) сложим 3 и 8, запишем сумму в первую ячейку: (11,8) — команда 1;

5) сложим 11 и 8, запишем сумму в первую ячейку: (19,8) — команда 1.

Таким образом, для получения пары (19,8) из (1,2) нужно использовать следующую последовательность команд: **12211**.

2-й способ (обратный).

Требуется получить из пары (19,8) пару (1,2). Так как не существует такого числа, сложив которое с 19, можно получить 8, последнее действие — получение числа 19 из 8 и неизвестного числа, т. е. $x + 8 = 19$; $x = 11$. Таким образом, пара (19,8) получена из (11,8) — команда 1.

Рассмотрим пару (11,8). Не существует такого положительного числа, сложив которое с 11, можно получить 8, следовательно, число 11 получено из 8 и неизвестного пока числа: $x + 8 = 11 \Rightarrow x = 3$. Значит, пара (11,8) получена из (3,8) — команда 1.

Рассмотрим пару (3,8). Не существует такого числа, сложив которое с 8, можно получить 3, следовательно, число 8 получено из 3 и неизвестного числа: $x + 8 = 3 \Rightarrow x = 5$. Таким образом, пара (3,8) получена из (3,5) (команда 2).

Рассмотрим пару (3,5). Число 5 было получено из числа 3 прибавлением к нему 2. Таким образом, пара (3,5) получена из (3,2) (команда 2).

Очевидно, наконец, что пара (3,2) получена из (1,2) — команда 1.

Запишем полученные команды в обратном порядке, начиная с последней: **22111**.

Ответ: 22111.

Задача 15. В формировании цепочки из четырех бусин используются следующие правила. В конце цепочки стоит одна из бусин P, N, T, O . На первом — одна из бусин P, R, T, O , которой нет на третьем месте. На третьем месте — одна из бусин O, P, T , не стоящая в цепочке последней. Определите, была ли создана цепочка $TTOO$ с учетом этих правил [5].

Решение.

Проанализируем заданную цепочку на соответствие указанным правилам. Цепочка $TTOO$ соответствует правилу №1, но не соответствует правилу №2 (так как на третьем месте не должно быть той же буквы, что и на первом).

Ответ: данная цепочка не была создана с учетом приведенных правил.

Задача 16. Витя пригласил своего друга Сергея в гости, но не сказал ему код от цифрового замка своего подъезда, а послал следующее SMS-сообщение: «В последовательности чисел 3, 1, 8, 2, 6 все числа больше 5 разделить на 2, а затем удалить из полученной последовательности все четные числа». Какой код получил Сергей, выполнив указанные в сообщении действия [7]?

Решение.

Выполним указанные в SMS-сообщении действия:

1) разделим все числа последовательности, большие 5, на 2:

3, 1, $8 / 2$, 2, $6 / 2 = 3$, 1, 4, 2, 3;

2) удалим из последовательности 3, 1, 4, 2, 3 четные числа: 3, 1, 3.

Таким образом, искомая последовательность — 3, 1, 3.

Ответ: 3, 1, 3.

Задачи для самостоятельного выполнения

Алгоритм. Исполнитель алгоритма

Задача 1. Строки (цепочки символов латинских букв) создаются по следующему правилу. Первая строка состоит из одного символа — латинской буквы А. Каждая из последующих цепочек создается такими действиями: в очередную строку сначала дважды подряд записывается предыдущая строка, а затем к ней справа приписывается буква, чей порядковый номер в алфавите соответствует номеру строки (на i -м шаге пишется i -я буква алфавита). Вот первые 4 строки, созданные по этому правилу:

- (1) А
- (2) ААВ
- (3) ААВААВС
- (4) ААВААВСААВААВСD

Латинский алфавит (для справки):
ABCDEFGHIJKLMNOPQRSTUVWXYZ.

Запишите шесть символов подряд, стоящие в восьмой строке со 101-го по 106-е место (считая слева направо).

Задача 2. Строки (цепочки символов латинских букв) создаются по следующему правилу. Первая строка состоит из одного символа — латинской буквы А. Каждая из последующих цепочек создается такими действиями: в очередную строку сначала дважды подряд записывается предыдущая строка, а затем к ней справа приписывается буква, чей порядковый номер в алфавите соответствует номеру строки (на i -м шаге пишется i -я буква алфавита). Вот первые 4 строки, созданные по этому правилу:

- (1) А
- (2) ААВ
- (3) ААВААВС
- (4) ААВААВСААВААВСD

Латинский алфавит (для справки):
ABCDEFGHIJKLMNOPQRSTUVWXYZ.

Сколько букв в восьмой строке отличается от буквы «В»?

Задача 3. Строки (цепочки символов латинских букв) создаются по следующему правилу. Первая строка состоит из одного символа — латинской буквы А. Каждая из последующих цепочек создается такими действиями: в очередную строку сначала дважды подряд записывается предыдущая строка, а затем к ней справа приписы-

вается буква, чей порядковый номер в алфавите соответствует номеру строки (на i -м шаге пишется i -я буква алфавита). Вот первые 4 строки, созданные по этому правилу:

- (1) A
- (2) AAB
- (3) AABAABC
- (4) AABAABCAABAABCD

Латинский алфавит (для справки):

ABCDEFGHIJKLMNOPQRSTUVWXYZ.

Запишите шесть символов подряд, стоящие в девятой строке с 315-го по 320-е место (считая слева направо).

Задача 4. Строки (цепочки символов латинских букв) создаются по следующему правилу. Первая строка состоит из одного символа — латинской буквы A. Каждая из последующих цепочек создается такими действиями: в очередную строку сначала дважды подряд записывается предыдущая строка, а затем к ней справа приписывается буква, чей порядковый номер в алфавите соответствует номеру строки (на i -м шаге пишется i -я буква алфавита). Вот первые 4 строки, созданные по этому правилу:

- (1) A
- (2) AAB
- (3) AABAABC
- (4) AABAABCAABAABCD

Латинский алфавит (для справки):

ABCDEFGHIJKLMNOPQRSTUVWXYZ.

Сколько букв в девятой строке отличается от буквы A?

Задача 5. Цепочки символов (строки) создаются по следующему правилу. Первая строка состоит из одного символа — цифры 1. Каждая из последующих цепочек создается следующим действием: в очередную строку дважды записывается предыдущая цепочка цифр (одна за другой, подряд), а в конец приписывается еще одно число — номер строки по порядку (на i -м шаге дописывается число i). Вот первые 4 строки, созданные по этому правилу:

- (1) 1
- (2) 112
- (3) 1121123
- (4) 112112311211234

Сколько цифр в восьмой строке отличается от цифры 8?

Задача 6. Упаковка информации методом RLE-кодирования состоит в следующем. Упакованная последовательность содержит управляющие байты, а за каждым управляющим байтом следует один или несколько байтов данных. Если старший бит управляющего байта равен 1, то следующий за управляющим байтом байт данных при распаковке нужно повторить столько раз, сколько записано в оставшихся 7 битах управляющего байта. Если же старший бит управляющего байта равен 0, то надо взять несколько следующих байтов данных без изменения, сколько именно — записано в оставшихся 7 битах управляющего байта. Например, управляющий байт 10000111 говорит о том, что следующий за ним байт надо повторить 7 раз, а управляющий байт 00000100 — о том, что следующие за ним 4 байта надо взять без изменений.

После кодирования методом RLE получилась следующая последовательность байтов (первый байт – управляющий):

10001011 11111111 00000011 10110011 00000001 11100010

Сколько байтов будет содержать данная последовательность после ее распаковки?

Задача 7. Упаковка информации методом RLE-кодирования состоит в следующем. Упакованная последовательность содержит управляющие байты, а за каждым управляющим байтом следует один или несколько байтов данных. Если старший бит управляющего байта равен 1, то следующий за управляющим байтом байт данных при распаковке нужно повторить столько раз, сколько записано в оставшихся 7 битах управляющего байта. Если же старший бит управляющего байта равен 0, то надо взять несколько следующих байтов данных без изменения, сколько именно — записано в оставшихся 7 битах управляющего байта. Например, управляющий байт 10000111 говорит о том, что следующий за ним байт надо повторить 7 раз, а управляющий байт 00000100 — о том, что следующие за ним 4 байта надо взять без изменений.

После кодирования методом RLE получилась следующая последовательность байтов (первый байт — управляющий):

10000111 10101010 00000001 10101111 00000010 10000101 10101010.

Сколько байтов будет содержать данная последовательность после ее распаковки?

Задача 8. Упаковка информации методом RLE-кодирования состоит в следующем. Упакованная последовательность содержит управляющие байты, а за каждым управляющим байтом следует

один или несколько байтов данных. Если старший бит управляющего байта равен 1, то следующий за управляющим байтом байт данных при распаковке нужно повторить столько раз, сколько записано в оставшихся 7 битах управляющего байта. Если же старший бит управляющего байта равен 0, то надо взять несколько следующих байтов данных без изменения, сколько именно — записано в оставшихся 7 битах управляющего байта. Например, управляющий байт 10000111 говорит о том, что следующий за ним байт надо повторить 7 раз, а управляющий байт 00000100 — о том, что следующие за ним 4 байта надо взять без изменений.

После кодирования методом RLE получилась следующая последовательность байтов (первый байт — управляющий):

**00000010 10101010 00000010 11111111 11111111 10000101
10101010.**

Сколько байтов будет содержать данная последовательность после ее распаковки?

Задача 9. Маша пригласила Таню в гости, но не сказала ей код от цифрового замка своего подъезда, а послала следующее SMS-сообщение: «В последовательности чисел 6, 2, 7, 9, 4 все числа меньше 5 разделить на 2, а затем удалить из полученной последовательности все четные числа». Определите, какой код получила Таня, выполнив указанные в сообщении действия.

Задача 10. Маша пригласила Таню в гости, но не сказала ей код от цифрового замка своего подъезда, а послала следующее SMS-сообщение: «В последовательности чисел 8, 4, 1, 7, 5, 3 из всех четных чисел вычесть 2, а затем удалить из полученной последовательности все числа больше 4». Определите, какой код получила Таня, выполнив указанные в сообщении действия.

Задача 11. Цепочка из трех бусин, помеченных буквами, формируется по следующему правилу. В конце цепочки стоит одна из бусин V, X, Y, Z . В середине — одна из бусин W, X, Z , которой нет на первом месте. На первом месте — одна из бусин W, X, Y , не стоящая на третьем месте. Соответствует ли цепочка WXX данному правилу?

Задача 12. Цепочка из трех бусин, помеченных буквами, формируется по следующему правилу. На третьем месте в цепочке стоит одна из бусин A, B, G . На втором — одна из бусин A, B, B . На первом месте — одна из бусин B, B, G , не стоящая в цепочке на втором или третьем месте. Соответствует ли цепочка GBG данному правилу?

Задача 13. При формировании цепочки из четырех бусин используются следующие правила. В начале цепочки стоит одна из бусин A, C, E, D . В конце — одна из бусин A, B, D, C , которой нет на втором месте. На втором месте — одна из бусин A, D, C , не стоящая в цепочке первой. Соответствует ли цепочка $DCEB$ данным правилам?

Задача 14. Для составления цепочек разрешается использовать бусины 5 типов, обозначаемых буквами N, S, O, P, E . Каждая цепочка должна состоять из четырех бусин, при этом должны соблюдаться следующие правила. На первом месте стоит одна из букв N, S, O . На третьем месте — любая согласная буква, если первая буква гласная, или любая гласная, если первая — согласная. На втором месте — одна из букв S, O, E , не стоящая на первом или третьем месте. На четвертом месте — любая гласная буква, не стоящая на втором или третьем месте. Соответствует ли цепочка $OESO$ данным правилам?

Задача 15. Для составления 4-значных чисел используются цифры 1, 2, 3, 4, 5. При этом соблюдаются следующие правила:

- 1) на первом месте стоит одна из цифр 3, 4 или 5;
 - 2) после каждой четной цифры идет нечетная, а после каждой нечетной — четная;
 - 3) последней цифрой не может быть цифра 2.
- Соответствует ли указанным правилам число 3454?

Задача 16. Исполнитель ЧЕРЕПАШКА перемещается на экране компьютера, оставляя след в виде линии. В каждый конкретный момент известно положение исполнителя и направление его движения. У исполнителя существуют две команды:

Вперед n , где n — целое число, вызывающая передвижение ЧЕРЕПАШКИ на n шагов в направлении движения;

Направо m , где m — целое число, вызывающая изменение направления движения на m° по часовой стрелке.

Запись **Повтори 5 [Команда1 Команда2]** означает, что последовательность команд в скобках повторится 5 раз. Исполнитель интерпретирует эту запись как одну команду.

ЧЕРЕПАШКЕ был дан для исполнения следующий алгоритм:
Повтори 3 [Повтори 3 [Вперед 60 Направо 120]]

Какая фигура появится на экране?

Задача 17. Исполнитель ЧЕРЕПАШКА перемещается на экране компьютера, оставляя след в виде линии. В каждый конкретный момент известно положение исполнителя и направление его движения. У исполнителя существуют две команды:

Вперед n , вызывающая передвижение ЧЕРЕПАШКИ на n шагов в направлении движения;

Направо m , вызывающая изменение направления движения на m° по часовой стрелке ($0 < m < 180$).

Вместо n и m должны стоять целые числа.

Запись **Повтори 5 [Команда1 Команда2]** означает, что последовательность команд в квадратных скобках повторится 5 раз.

Какое число нужно записать вместо m в алгоритме **Повтори 5 [Вперед 40 Направо m]**, чтобы на экране появился правильный пятиугольник?

Задача 18. Исполнитель РОБОТ действует на клетчатой доске, между соседними клетками которой могут стоять стены.

РОБОТ передвигается по клеткам доски и может выполнять команды: 1 — *вверх*, 2 — *вниз*, 3 — *вправо*, 4 — *влево*, переходя на соседнюю клетку в указанном направлении. Если в этом направлении между клетками стоит стена, то РОБОТ разрушается.

РОБОТ успешно выполнил программу 33233241. Какую последовательность из четырех команд должен выполнить РОБОТ, чтобы вернуться в ту же самую клетку, где он был перед началом выполнения программы, и не разрушиться вне зависимости от того, какие стены стоят на поле?

Задача 19. Система команд исполнителя РОБОТ, «живущего» в прямоугольном лабиринте на клетчатой плоскости: *вверх*, *вниз*, *влево*, *вправо*. При выполнении этой команды РОБОТ перемещается на соответствующую клетку.

Команды проверки истинности условия на наличие стены у той клетки, где он находится: *сверху свободно*, *снизу свободно*, *слева свободно*, *справа свободно*. Если РОБОТ начнет движение в сторону стены, то он разрушится.

Какие клетки данного лабиринта соответствуют требованию, что, выполнив предложенную программу, РОБОТ остановится в той же самой клетке, с которой он начал движение?

НАЧАЛО

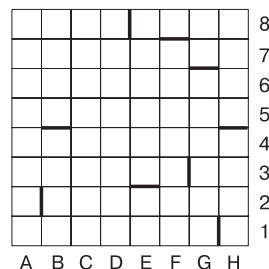
ПОКА < *снизу свободно* > *вниз*

ПОКА < *справа свободно* > *вправо*

ПОКА < *сверху свободно* > *вверх*

ПОКА < *слева свободно* > *влево*

КОНЕЦ



Задача 20. Система команд исполнителя РОБОТ, «живущего» в прямоугольном лабиринте на клетчатой плоскости: *вверх*, *вниз*, *влево*, *вправо*. При выполнении этих команд РОБОТ перемещается на одну клетку соответственно: *вверх* ↑, *вниз* ↓, *влево* ←, *вправо* →.

Четыре команды проверяют истинность условия отсутствия стены у той клетки, где находится РОБОТ: *сверху свободно*, *снизу свободно*, *слева свободно*, *справа свободно*.

Если РОБОТ начнет движение в сторону стены, то он разрушится.

Какие клетки данного лабиринта соответствуют требованию, что, выполнив предложенную программу, РОБОТ остановится в той же самой клетке, с которой он начал движение?

НАЧАЛО

ПОКА < *снизу свободно* > *вниз*

ПОКА < *справа свободно* > *вправо*

ПОКА < *сверху свободно* > *вверх*

ПОКА < *слева свободно* > *влево*

КОНЕЦ

								8
								7
								6
								5
								4
								3
								2
								1
A	B	C	D	E	F	G	H	

Задача 21. У исполнителя КАЛЬКУЛЯТОР две команды, которым присвоены номера:

1 — *прибавь 7*;

2 — *умножь на 5*.

Выполняя первую из них, КАЛЬКУЛЯТОР прибавляет к числу на экране 7, а выполняя вторую, умножает его на 5. Запишите порядок команд в программе получения из 0 числа 910, содержащей не более 5 команд, указывая лишь номера этих команд.

Задача 22. У исполнителя КАЛЬКУЛЯТОР две команды, которым присвоены номера:

1 — *прибавь 1*;

2 — *умножь на 3*.

Выполняя первую из них, КАЛЬКУЛЯТОР прибавляет к числу на экране 1, а выполняя вторую, умножает его на 3. Запишите порядок команд в программе получения из 3 числа 91, содержащей не более 5 команд, указывая лишь номера этих команд.

Задача 23. Система команд исполнителя ПРЯМОУГОЛЬНИК:

1) *длина* — уменьшает длину прямоугольника вдвое;

2) *ширина* — уменьшает ширину прямоугольника на треть.

Известно, что после выполнения алгоритма **ПОВТОРИ 5 (длина ширина)** площадь прямоугольника оказалась равной 12 см^2 , а после выполнения алгоритма **ПОВТОРИ 2 (длина ширина)** ширина прямоугольника стала равной 9 см. Найдите исходную длину прямоугольника.

Задача 24. По бесконечному клетчатому листу бумаги с привязанной к нему системой координат $ХОУ$ из некоторого начального положения начинает скакать **КУЗНЕЧИК**. Он одним скачком может совершать следующие действия:

- 1) увеличить свою координату x на 2;
- 2) уменьшить свою координату x на 2;
- 3) увеличить свою координату y на 2;
- 4) уменьшить свою координату y на 2;

Последовательность действий, которые он выполнил, такова: 131322. Каково наименьшее число скачков, приводящих **КУЗНЕЧИКА** из прежнего начального положения в то же самое конечное?

Задача 25. Исполнитель **КАЛЬКУЛЯТОР**, разработанный школьниками на уроках программирования, имеет три регистра (их имена: A, B, C) и позволяет выполнять с регистрами поразрядные операции. Система команд этого исполнителя:

- 1) $>A$ — ввод данных в регистр A ;
- 2) $<A$ — вывод данных из регистра A ;
- 3) $A + B$ — сохранить без изменения нулевые разряды регистра B , соответствующие единичным разрядам регистра A , а остальные разряды регистра B инвертировать.

Определите функцию, которая вычисляется следующей программой:

$>A >B C+A B+C A+B C+A C+C <C$

Задача 26. Исполнитель **КАЛЬКУЛЯТОР**, разработанный школьниками на уроках программирования, имеет три регистра (их имена: A, B, C) и позволяет выполнять с регистрами поразрядные операции. Система команд этого исполнителя:

- 1) $>A$ — ввод данных в регистр A ;
- 2) $<A$ — вывод данных из регистра A ;
- 3) $A + B$ — сохранить без изменения нулевые разряды регистра B , соответствующие единичным разрядам регистра A , а остальные разряды регистра B инвертировать.

Определите функцию, которая вычисляется следующей программой:

$>C >A C+A C+B A+B B+A <A$

Задача 27. Исполнитель ПРОЦЕССОР имеет два регистра с именами A и B , в которых хранятся целые числа. В систему команд ПРОЦЕССОРА входят следующие команды:

$>An$	Ввод числа n в ячейку A
$>Bn$	Ввод числа n в ячейку B
$<A$	Вывод данных из ячейки A на экран
$<B$	Вывод данных из ячейки B на экран
$*BA$	Перемножить содержимое ячейки B на содержимое ячейки A и полученный результат поместить в ячейку B . Содержимое ячейки A остается неизменным
$/BA$	Разделить нацело содержимое A на содержимое ячейки B и полученный результат поместить в ячейку A . Содержимое ячейки B остается неизменным
$*AB$	Перемножить содержимое ячейки B на содержимое ячейки A и полученный результат поместить в ячейку A . Содержимое ячейки B остается неизменным
$/AB$	Разделить нацело содержимое ячейки A на содержимое ячейки B и полученный результат поместить в ячейку A . Содержимое ячейки B остается неизменным

Определите числа, которые будут выведены на экран после выполнения команд

$>A2 >B3 /BA *AB *BA /AB <A <B$

Задача 28. В приведенном фрагменте алгоритма, записанном на алгоритмическом языке, тип переменных a, b, c — «строка», а переменных i, k — «целое». Используются следующие функции:

- 1) Длина(a) — возвращает количество символов в строке a (тип «целое»);
- 2) Извлечь(a, i) — возвращает i -й (слева) символ в строке a (тип «строка»);
- 3) Склеить(a, b) — возвращает строку, в которой записаны сначала все символы строки a , а затем все символы строки b (тип «строка»).

Значения строк записываются в одинарных кавычках.

Фрагмент алгоритма:

```
i := Длина (a)
b := Извлечь (a, i - 4)
c := Извлечь (a, i - 5)
b := Склеить (b, c)
c := Извлечь (a, i)
b := Склеить (b, c)
```

```
с := Извлечь (а, i - 6)
b := Склеить (b, с)
i := 5
пока i > 0
нц
    с := Извлечь (а, i)
    b := Склеить (с, b)
    i := i - 1
кц
```

Какое значение будет у переменной *b* после выполнения приведенного фрагмента алгоритма, если значение переменной *a* было "Электроника"?

Глава 5

Введение в программирование на языке Паскаль

5. 1. Основные понятия языка Паскаль

Алфавит языка программирования Паскаль включает в себя буквы, цифры, символы и зарезервированные слова:

1) буквы — латинские от **a** до **z** (различий между прописными и строчными буквами нет) и символ подчеркивания (**_**);

2) цифры — арабские от **0** до **9** и шестнадцатеричные (первые 10 цифр от **0** до **9** — арабские, остальные шесть — латинские буквы: **a, b, c, d, e, f**);

3) символы — **+ - * / = , . : ; < > [] () { } ' , \$**, а также пары символов **< > < = > = := (* *)** и символ пробела;

4) зарезервированные слова — **abs, and, array, begin, case, const, dir, do, downto, else, end, for, function, goto, if, int, label, mod, not, of, or, procedure, program, repeat, shr, then, to, type, var, while, with** и др.

Программа на языке Паскаль состоит из:

1) названия программы, начинающегося с зарезервированного слова **program** и имени, состоящего из букв латинского алфавита и цифр;

2) раздела перечисления модулей, используемых в программе, который начинается с ключевого слова **uses**, а затем через запятую перечисляются имена модулей;

3) раздела определения меток, который начинается с зарезервированного слова **label**;

4) раздела определения констант, который начинается с зарезервированного слова **const**;

5) раздела определения типов, который начинается с зарезервированного слова **type**;

6) раздела описания переменных, который начинается с зарезервированного слова **var**, а затем следует перечисление используемых в программе переменных с указанием их типов;

7) раздела описания процедур и функций, который начинается с зарезервированного слова **procedure** или **function** соответственно;

8) раздела операторов, который начинается с зарезервированного слова **begin** и заканчивается зарезервированным словом **end**

со стоящей после него точкой (**end.**). Этот раздел включает команды, которые должен выполнить исполнитель (компьютер). Зарезервированные слова **begin** и **end** отмечают начало и конец программы. Текст, находящийся после слова **end** с точкой, компилятором игнорируется.

Таким образом, *структура программы* на языке Паскаль выглядит следующим образом:

```
program <имя_программы>;  
[uses <имена_подключаемых_модулей>;]  
[label <список_меток>;]  
[const <имя_константы> = <значение_константы>;]  
[type <имя_типа> = <определение_типа>;]  
[var <имя_переменной>: <тип_переменной>;]  
[procedure <имя_процедуры> <описание_процедуры>;]  
[function <имя_функции> <описание_функции>;]  
begin    {начало основного тела программы}  
    <операторы>;  
end.    {конец основного тела программы}
```

Выше в угловых скобках указаны части программы, которые должен написать программист. В квадратных скобках указаны разделы, которые в какой-то программе могут отсутствовать. Текст же в фигурных скобках — {...} — представляет собой *комментарии* и компьютером игнорируется.

Порядок следования разделов программы может быть любым, но главное, чтобы использование того или иного объекта выполнялось после его объявления.

Обрабатываемые данные в ЭВМ представляются в виде величин и их совокупностей, где под *величиной* понимается некоторый элемент данных (число, символ, набор символов и т. п.).

Величина описывается следующим набором параметров: *имя*, *тип* (определяющий множество допустимых значений и применимых операций) и *значение*. В программе можно использовать переменные и постоянные величины. Постоянной (*константой*) называется величина, значение которой указывается в тексте программы и не изменяется в процессе ее исполнения. *Переменной* называется величина, значение которой изменяется в процессе исполнения программы.

Имена, назначаемые программным объектам (константам, типам, переменным, функциям, процедурам и всей программе в целом), называют *идентификаторами*. Они могут состоять только из латин-

ских букв, цифр и знака подчеркивания («_»), причем имя не может начинаться с цифры. В качестве имени также нельзя использовать зарезервированные слова.

Переменные, используемые в программе, должны быть описаны в разделе **var** по следующему шаблону:

```
var  
  <имя_переменной>: <имя_типа>;
```

Имена переменных одного и того же типа можно перечислить через запятую, указав после двоеточия их тип. Переменные разных типов описываются отдельно и отделяются друг от друга точкой с запятой.

Константы описываются в разделе **const** по следующему шаблону:

```
const  
  <имя_константы> = <значение>;
```

В качестве констант могут использоваться числа, символы, строки и множества. Пример объявления констант:

```
const  
  pi = 3.14;  
  ten = 10;
```

5.2. Типы данных языка Паскаль

Типы данных можно разделить на базовые, символьные и структурированные.

Базовые типы делятся на порядковые и вещественные, где *порядковые типы* — это целый, логический, перечисляемый и интервальный типы.

К *структурированным типам* относятся массив, файл, запись, множество.

Тип данных определяет возможный диапазон значений величины, относящийся к этому типу, а также операции, допустимые над величиной этого типа.

Каждому элементу порядкового типа сопоставлен уникальный (порядковый) номер. Нумерация значений при этом начинается с нуля, за исключением типов **shortint** и **longint**.

Целочисленные типы данных приведены в табл. 5.1.

Над целочисленными величинами определены следующие операции: сложение (+), вычитание (−), умножение (*), деление (/), деление нацело (div) и определение остатка от деления нацело (mod).

Таблица 5.1

Целочисленные типы данных

Тип данных	Количество байтов	Диапазон значений	
shortint	1	-128..127	$-2^7..2^7-1$
byte	1	0..255	$0..2^8-1$
word	2	0..65535	$0..2^{16}-1$
longint	4	-2147483648..2147483647	$-2^{31}..2^{31}-1$

Величина логического типа (boolean) может принимать два значения: false (ложь) или true (истина), например:

```
var
  flag: boolean;
```

Функции и процедуры для обработки величин, относящихся к порядковым типам данных, приведены в табл. 5.1.

Таблица 5.2

Функции и процедуры для работы с величинами порядкового типа

Имя	Описание	Пример использования
<i>Функции</i>		
ord(x)	Возвращает порядковый номер значения переменной <i>x</i>	ord('c') = 99
pred(x)	Возвращает значение, предшествующее <i>x</i> (к первому элементу неприменима)	pred('c') = 'b'
succ(x)	Возвращает значение, следующее за <i>x</i> (к последнему элементу неприменима)	succ('c') = 'd'
<i>Процедуры</i>		
inc(x)	Возвращает значение, следующее за <i>x</i>	inc('c') = 'd' inc(1) = 2
inc(x, k)	Возвращает <i>k</i> -е значение, следующее за <i>x</i> (для арифметических типов данных это эквивалентно оператору <i>x</i> := <i>x</i> + <i>k</i>)	inc('c', 2) = 'e' inc(1, 2) = 3
dec(x)	Возвращает значение, предшествующее <i>x</i>	dec('c') = 'b' dec(1) = 0
dec(x, k)	Возвращает <i>k</i> -е значение, предшествующее <i>x</i>	dec('c', 2) = 'a' dec(1, 2) = -1

Вещественные числа могут быть записаны в форме с фиксированной или плавающей точкой (табл. 5.3).

Таблица 5.3

Вещественные типы данных

Тип	Количество байт	Диапазон значений
single	4	$-1.5 \cdot 10^{45} \dots 3.4 \cdot 10^{38}$
double	8	$-5.0 \cdot 10^{324} \dots 1.7 \cdot 10^{308}$
extended	10	$-3.4 \cdot 10^{4932} \dots 1.1 \cdot 10^{4932}$
comp	8	$-2^{63}+1 \dots 2^{63}-1$

Над вещественными величинами допустимы следующие операции: сложение (+), вычитание (−), умножение (*) и деление (/).

Тип результата операций сложения, вычитания и умножения зависит от типов операндов:

Операнд 1	Операнд 2	Результат
целое	целое	целое
целое	вещественное	вещественное
вещественное	целое	вещественное
вещественное	вещественное	вещественное

Для операции деления тип результата, независимо от типов операндов, всегда будет вещественным. Разделитель целой и дробной частей в вещественном числе — символ «.» (точка).

В целом в Паскале определены следующие операции:

- 1) унарные — not (отрицание);
- 2) мультипликативные — * (умножение), / (деление), div (деление нацело), mod (остаток от целочисленного деления), and (логическое «И»);
- 3) аддитивные — + (сложение), − (вычитание), or (логическое «ИЛИ»), xor (исключающее «ИЛИ»);
- 4) отношения — = (равно), <> (не равно), < (меньше), > (больше), <= (меньше или равно), >= (больше или равно).

Приоритет этих операций убывает в том порядке, в каком они приведены в списке. При равном же приоритете операции выполняются слева направо в порядке их записи в этом списке. Для изменения очередности выполнения операций следует использовать круглые скобки.

Для вычисления наиболее распространенных математических функций в Паскале предусмотрены *стандартные функции*, приведенные в табл. 5.4.

Таблица 5.4

Математические функции

Имя функции	Описание	Возвращаемый результат
<code>sin(x)</code>	Вычисление синуса угла x (x должен быть представлен в радианах)	Вещественный
<code>cos(x)</code>	Вычисление косинуса угла x (x должен быть представлен в радианах)	Вещественный
<code>arctan(x)</code>	Вычисление арктангенса угла x (x должен быть представлен в радианах)	Вещественный
<code>ln(x)</code>	Вычисление натурального логарифма числа x	Вещественный
<code>exp(x)</code>	Вычисление экспоненты числа x (e^x)	Вещественный
<code>sqr(x)</code>	Вычисление квадрата числа x (x^2)	Зависит от типа аргумента
<code>sqrt(x)</code>	Вычисление квадратного корня числа x	Вещественный
<code>abs(x)</code>	Вычисление абсолютной величины числа x	Зависит от типа аргумента
<code>trunc(x)</code>	Отбрасывание дробной части числа x	Целый
<code>frac(x)</code>	Получение дробной части числа x	Вещественный
<code>int(x)</code>	Получение целой части числа x	Вещественный
<code>round(x)</code>	Округление числа x к ближайшему целому	Целый
<code>power(x, y)</code>	Возведение числа x в степень y (для использования необходимо подключить модуль <code>Math: uses Math;</code>)	Вещественный
<code>random(x)</code>	Генерация случайного числа на отрезке $[0; x - 1]$	Вещественный
<code>random</code>	Генерация случайного числа на полуинтервале $[0; 1)$	Вещественный

5.3. Операторы языка Паскаль

5.3.1. Основные операторы языка Паскаль

Оператором называется минимальная единица программы. Все операторы языка Паскаль должны заканчиваться знаком «;» (точка с запятой), и ни один оператор не может разрываться этим знаком.

К основным операторам языка Паскаль относятся:

1) оператор присваивания ($a := b;$) — используется для присваивания переменной a значения другой переменной b . В правой части оператора присваивания может находиться переменная, константа, арифметическое выражение или вызов функции (процедуры);

2) пустой оператор « $;$ » (символ точки с запятой);

3) операторные скобки (составной оператор), превращающие несколько операторов в один*:

```
begin
    <оператор>;
end;
```

5.3.2. Операторы ввода данных

Ввод данных в программу основан на использовании процедур `read` и `readln`. Они позволяют считывать данные, вводимые пользователем с клавиатуры, и сохранять их в переменные. Переменные, в которые выполняется запись данных, указываются в качестве параметров этих процедур в круглых скобках через запятую. С помощью команд `read` и `readln` можно вводить значения только базовых типов, а также строки и символы.

После того как процедура `read` (или `readln`) получает управление, на экране появляется окно**, в которое требуется ввести значения переменных. Вводимые значения необходимо при этом разделять пробелами, а завершать ввод нажатием клавиши **Enter**. Ввод данных закончится в тот момент, когда последняя переменная, указанная в параметрах процедуры, получит свое значение.

Различие процедур `read` и `readln` состоит в том, что при выполнении нескольких подряд процедур `read` вводимые ими значения считываются из одной и той же строки ввода. Процедура же `readln` считывает заданное количество значений переменных с текущей строки, игнорирует оставшиеся в этой строке значения и переводит курсор на новую строку.

Рассмотрим программу **TestInputOne**:

```
program TestInputOne;
var
    a, b: byte;
```

* Здесь и далее там, где в записи конструкций языка Паскаль будет использовано обозначение `<оператор>`, его следует понимать как «один оператор или несколько операторов, заключенные в операторные скобки `begin .. end`».

** В некоторых версиях Pascal-трансляторов (Free Pascal, Turbo Pascal и др.) ввод данных может производиться просто в очередной экранной строке. — *Прим. ред.*

```
begin
  readln(a, b);
end.
```

Предположим, что после запуска этой программы введены следующие данные: **12 5 6.6 text**. Тогда в результате работы процедуры `readln` в переменную *a* будет записано значение 12, в переменную *b* — значение 5, а оставшиеся данные (**6.6 text**) будут проигнорированы.

Рассмотрим программу **TestInputTwo**:

```
program TestInputTwo;
var
  a, b, c, d: byte;
begin
  read(a, b, c);
end.
```

Пусть пользователь ввел следующие данные: **12 5 6.6**. В результате работы процедуры `read` в переменную *a* также будет занесено значение 12, а в *b* — значение 5. Но затем работа этой программы завершится отказом исполнителя, так как будет произведена попытка записать в целочисленную переменную *c* вещественное значение 6.6.

Примечание. Использование процедуры `readln` без параметров позволяет приостановить выполнение программы до нажатия клавиши **Enter**.

5.3.3. Операторы вывода данных

Вывод значений величин (только базового и символьного типов), описанных в программе, осуществляется с помощью процедур `write` и `writeln`. Каждый параметр такой процедуры соответствует величине (константе или переменной), значение которой выводится на экран; параметры отделяются друг от друга запятыми. Чтобы вывести константу — строку символов, необходимо заключить ее в одиночные кавычки и указать в качестве одного из параметров.

Дополнительно при этом можно определить следующие параметры оператора вывода:

- 1) ширину поля вывода (для целочисленных и вещественных величин);
- 2) количество выводимых цифр после десятичной точки (для вещественных чисел).

Формат записи процедуры `write`:

```
write(arg:P:Q),
```

где *arg* — выводимая величина, *P* — ширина поля, а *Q* — длина дробной части.

Например, в результате работы программы:

```
program WriteExample;  
var  
  a, b, c: single;  
begin  
  a := 3.1;  
  b := -2.3;  
  c := 1.5;  
  write('A=', A:4:1, ' B=', B, ' C=', C:6:1);  
  readln;  
end.
```

на экран будет выведено:

```
A= 3.1 B=-2.299999952E+00 C= 1.5
```

При недостатке количества позиций для вывода число или текст выводятся полностью, а формат игнорируется (кроме формата вывода дробной части числа). Если же формат вывода не задан, то значения целых и строковых переменных выводятся полностью, а вещественных — в экспоненциальной форме.

Различие в работе процедур `write` и `writeln`: процедура `writeln` после вывода данных переводит курсор на следующую строку, в результате чего вывод последующих данных происходит уже на новой строке, а `write` оставляет курсор на той же строке. Использование же оператора `writeln` без параметров выполняет перевод строки.

Например, результат выполнения программы **WritelnExample**:

```
program WritelnExample;  
var  
  z, x: byte;  
begin  
  x := 2;  
  z := 4;  
  write('z=', z);  
  writeln; {*}  
  writeln('x=', x);  
  readln;  
end.
```

будет следующим:

```
z=4  
x=2
```

При отсутствии же оператора `writeln`, помеченного комментарием `{*}`, мы получим другой вывод:

```
z=4x=2.
```

Использование оператора `writeln` с соответствующим текстом перед оператором `read (readln)` позволяет напомнить пользователю, какие данные нужно ввести.

Примечание. Очистку экрана можно выполнить с помощью функции `clrscr`. Для подключения модуля, в котором описана данная функция, нужно использовать команду `uses Crt`.

Задача 1. Определите значение целочисленных переменных a и b после выполнения фрагмента программы:

```
a := 1819;  
b := (a div 100)*10+9;  
a := (10*b-a) mod 100;
```

(`div` и `mod` — операции, вычисляющие результат деления нацело первого аргумента на второй и остаток от деления соответственно) [4].

Решение.

Значения переменных a и b вычисляются в три этапа:

- 1) $a := 1819$;
- 2) $b := (1819 \text{ div } 100) * 10 + 9 = 18 * 10 + 9 = 180 + 9 = 189$;
- 3) $a := (10 * 189 - 1819) \text{ mod } 100 = (1890 - 1819) \text{ mod } 100 = 71 \text{ mod } 100 = 71$.

Таким образом, после выполнения данного фрагмента программы значение переменной a равно 71, а значение переменной b равно 189.

Ответ: $a = 71, b = 189$.

5.4. Перечисляемый тип

Перечисляемый тип задается набором значений, которые могут принимать величины данного типа:

type

```
<имя типа> = (<идентиф. 1>, ..., <идентиф. n>);
```

где `<идентиф. 1>, ..., <идентиф. n>` — значения типа (идентификаторы), указанные программистом. (Условия, налагаемые на имена идентификаторов, приведены в разделе 5.1 данной книги.) К величинам перечисляемого типа применимы только операции отношения и операторы ввода-вывода.

Пример определения перечисляемого типа:

```
type
  TSeason = (Winter, Spring, Summer, Autumn);
  TDayOfWeek = (Mon, Tue, Wed, Thu, Fri, Sat, Sun);
```

Переменную перечисляемого типа можно объявить двумя способами:

1) описав переменную в разделе **var** и указав ее возможные значения:

```
var
  <имя переменной>: (<идентиф. 1>, ..., <идентиф. n>);
```

Пример:

```
var
  Nums: (one, two, three);
```

2) используя при описании переменной ранее определенный тип:

```
var
  Days: TDayOfWeek;
```

В программе **EnumerationType** показан пример использования перечисляемого типа:

```
program EnumerationType;
type
  TDayOfWeek = (Mon, Tue, Wed, Thu, Fri, Sat, Sun);
var
  d1, d2: TDayOfWeek;
  Season: (Winter, Spring, Summer, Autumn);
begin
  write('Введите название сезона: ');
  readln(Season);
  writeln('Сезон: ', Season);
  write('Введите первый день: ');
  readln(d1);
  writeln('Первый день: ', d1);
  write('Введите второй день: ');
  readln(d2);
  writeln('Второй день: ', d2);
  writeln(d1, ' > ', d2, ' = ', d1 > d2);
  readln;
end.
```

Результат работы программы:

```
Введите название сезона: spring
Сезон: Spring
Введите первый день: fri
Первый день: Fri
Введите второй день: mon
Второй день: Mon
Fri > Mon = TRUE
```

5.5. Интервальный тип

Интервальный тип задается диапазоном значений, которые могут принимать величины данного типа. Для описания интервального типа следует указать начальное значение, поставить две точки и указать конечное значение:

```
type
    <имя типа> = <начальное значение> .. <конечное
    значение>;
```

Пример:

```
type
    THours = 1..24;
```

Переменную интервального типа можно объявить двумя способами:

1) объявив переменную в разделе **var** и указав диапазон ее значений:

```
var
    <имя переменной>: <начальное значение> .. <конечное
    значение>;
```

Пример:

```
var
    VHours: 1..24;
```

2) используя при описании переменной ранее определенный тип:

```
var
    VHours: THours;
```

При определении интервального типа можно использовать перечисляемый тип, целые и символьные типы, *но не вещественные*.

Пример программы:

```
program SubRangeType;
var
  VMin: 1..60;
  VHours: 1..24;
  VChar: 'f'..'z';
begin
  VMin := 20;
  VHours := 11;
  VChar := 't';
  writeln(VChar, VHours, ': ', VMin);
  readln;
end.
```

5.6. Условный оператор

Условный оператор используется для организации *ветвления* алгоритма, т. е. для выполнения определенных действий в зависимости от истинности или ложности заданного условия. Условный оператор записывается следующим образом:

```
if <условие> then
  <оператор 1>
else
  <оператор 2>;
```

Если <условие> истинно, то выполняется <оператор 1>, иначе — <оператор 2>. В качестве <условия> при этом может выступать любое выражение, результат вычисления которого принимает значение `true` или `false`. (Слова **if**, **then** и **else** являются зарезервированными. Перед **else** точка с запятой («;») не ставится.) При записи условного оператора возможно использование различных логических операторов.

В качестве примера рассмотрим программу **IfExampleFull**:

```
program IfExampleFull;
var
  x, y, a, b: single;
begin
  a := 2.4;
  b := -1.2;
```

```
write('Введите x = ');  
readln(x);  
if x <= 0 then  
    y := -a*x  
else  
    y := b*exp(x);  
writeln('y = ', y:6:3);  
readln;  
end.
```

После ввода значения переменной x с помощью условного оператора анализируется ее значение. Если введенное значение x меньше или равно нулю, то y будет вычислено следующим образом: $y := -a*x$. В противном случае (при $x > 0$) значение y будет вычислено как $y := b*\exp(x)$;

Существует также сокращенная форма условного оператора (без части, связанной с **else**). При использовании сокращенной формы оператор, следующий за **then**, выполняется, если это условное выражение истинно (см. программу **IfExampleTrunc**); в противном случае просто управление передается дальше, на первый оператор после условного оператора. Пример программы:

```
program IfExampleTrunc;  
var  
    x, y, a, b: single;  
begin  
    a := 2.4;  
    b := -1.2;  
    write('Введите x = ');  
    readln(x);  
    if x <= 0 then  
        y := -a*x;  
    writeln('y = ', y:4:2);  
    readln;  
end.
```

Блок-схема условного оператора в полной и в сокращенной форме показана на рис. 5.1. Ромб на этом рисунке обозначает блок проверки условия и может содержать две исходящие стрелки, которые указывают на операторы, выполняемые в зависимости от истинности или ложности условия:

1) если результат вычисления равен `true`, то выполняется <Оператор 1>;

2) если результат вычисления равен `false`, то выполняется <Оператор 2>.

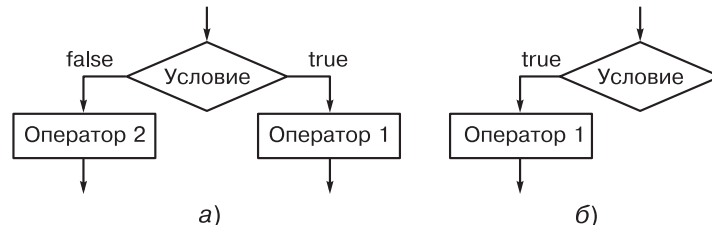


Рис. 5.1. Блок-схема условного оператора:
а) полная форма; б) сокращенная форма

Если при выполнении условия должно выполняться несколько операторов, то эти операторы необходимо объединить с помощью операторных скобок **«begin .. end»**.

Задача 2. Определите значение переменной *c* после выполнения следующего фрагмента программы [7]:

```

a := 100;
b := 30;
a := a - b * 3;
if a > b then
    c := a - b
else
    c := b - a;

```

Решение.

Рассмотрим выполнение программы по шагам:

1) в переменную *a* записывается значение 100;

2) переменной *b* присваивается значение 30;

3) вычисляется значение переменной *a*:

$a := a - b * 3$, т.е. $a := 100 - 30 * 3 = 10$;

4) с помощью условного оператора вычисляется значение переменной *c*: так как значение выражения $a > b$ ($10 > 30$) есть ложь, то значение *c* вычисляется с помощью оператора, записанного в ветви **else**: $c := b - a$, т.е. $c := 30 - 10 = 20$.

Ответ: $c = 20$.

При разработке программ возможно использование *вложенных* условных операторов (рис. 5.2):

```

if <условие 1> then
    if <условие 2> then
        <оператор 1>
    else
        <оператор 2>
else
    <оператор 3>;

```

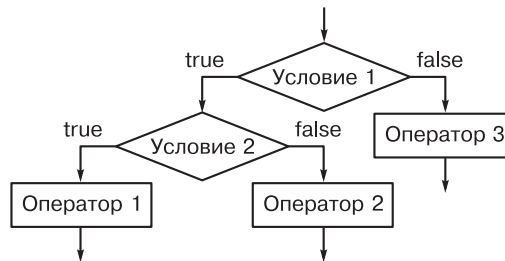


Рис. 5.2. Блок-схема вложенных условных операторов

Задача 3. Разработать программу вычисления значения переменной y в соответствии с блок-схемой, показанной на рис. 5.3, а.

Решение.

Из рис. 5.3, а видно, что для написания программы необходимо использовать два условных оператора, один из которых вложен в другой, причем первый из них — в полной, а второй — в сокращенной форме.

Рассмотрим особенности использования вложенных условных операторов.

Компилятор каждому **else** ставит в соответствие ближайший предшествующий **if**. Допустим, что по заданной блок-схеме написана программа **NestedIfFalse**:

```

program NestedIfFalse;
var
    x, y: byte;
begin
    write('Введите x = ');
    readln(x);

```

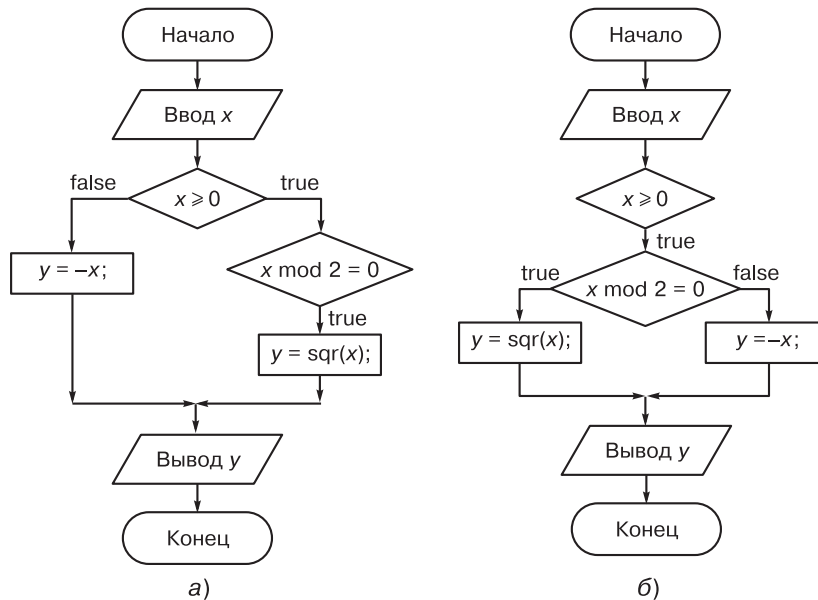



Рис. 5.3. Блок-схемы: а) программы **NestedIfBE** (**NestedIfElse**); б) программы **NestedIfFalse**

```

if x >= 0 then
    if x mod 2 = 0 then
        y := sqr(x)
    else
        y := -x;
    writeln('y = ', y);
    readln;
end.
  
```

В этой программе ветвь **else** относится к вложенному условному оператору, и такая программа будет выполняться не по блок-схеме, показанной на рис. 5.3, а, а по блок-схеме, показанной на рис. 5.3, б, т. е. логика работы программы будет неверной.

Существует два способа правильной записи программы:

- 1) заключить вложенный условный оператор в операторные скобки «**begin** .. **end**» (см. программу **NestedIfBE**);
- 2) дописать ветвь **else** вложенного условного оператора, в которой будут отсутствовать операторы (см. программу **NestedIfElse**).

```

program NestedIfBE;
var
    x, y: longint;
begin
    write('Введите x = ');
    readln(x);
    y := -1;
    if x >= 0 then
        begin
            if x mod 2 = 0 then
                y := sqr(x)
            end
        else
            y := -x;
            writeln('y = ', y);
            readln;
        end.

```

```

program NestedIfElse;
var
    x, y: longint;
begin
    write('Введите x = ');
    readln(x);
    y := -1;
    if x >= 0 then
        if x mod 2 = 0 then
            y := sqr(x)
        else
            y := -x;
            writeln('y = ', y);
            readln;
        end.

```

Задача 4. Ввести значение переменной y (y — величина угла в радианах). Если введенное значение переменной y четное и положительное, то вычислить значение x по формуле: $x := \sin(y) + \cos(y)$, иначе вывести сообщение об ошибке.

Решение.

Условие, указанное в задаче, можно записать следующим образом:

```
if (y mod 2 = 0) and (y > 0) then ...
```

Оператор **and** используется для проверки, отвечает ли введенное значение y условию четности ($y \bmod 2 = 0$) и положительности ($y > 0$). Если оба этих условия истинны, то выводится вычисленное значение x , в противном случае выводится сообщение об ошибке (см. программу **ComplexIf**).

```

program ComplexIf;
var
    x: single;
    y: word;
begin
    write('Введите y = ');
    readln(y);

```

```

if (y mod 2 = 0) and (y > 0) then
begin
  x := sin(y) + cos(y);
  writeln('x = ', x);
end
else
  writeln('Ошибка!');
readln;
end.

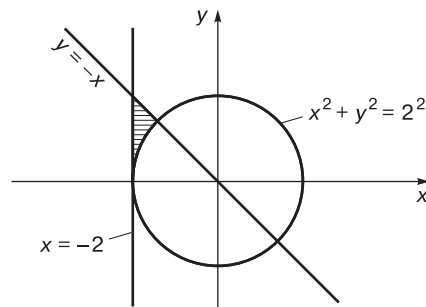
```

Задача 5. Требовалось написать программу, которая вводит с клавиатуры координаты точки плоскости (x, y — действительные числа) и определяет принадлежность точки заштрихованной области, включая ее границы. Программист торопился и написал программу неправильно:

```

program TestXY;
var
  x, y: single;
begin
  readln(x, y);
  if x * x + y * y >= 4 then
    if x >= -2 then
      if y <= -x then
        write('принадлежит')
      else
        write('не принадлежит');
    end.
end.

```



Последовательно выполните следующее:

1) приведите пример таких чисел x, y , при которых программа неверно решает поставленную задачу;

2) укажите, как нужно доработать программу, чтобы не было случаев ее неправильной работы [7].

Решение.

Проверка принадлежности точки заштрихованной области в программе выполняется с помощью следующих условий:

- 1) $x * x + y * y \geq 4$;
- 2) $x \geq -2$;
- 3) $y \leq -x$.

Однако в приведенной в задаче программе не проверяется условие $y \geq 0$. Поэтому если ввести отрицательное значение переменной y , то программа правильно работать не будет.

Допустим, что введены следующие значения переменных: $x = 2$, $y = -2$. В этом случае условия $x * x + y * y \geq 4$ и $x \geq -2$ истинны; истинно и условие $y \leq -x$. Следовательно, на экране появится надпись «принадлежит». Однако на самом деле точка $(2, -2)$ выделенной области не принадлежит!

Для проверки можно ввести координаты $(2, -3)$. В этом случае, как несложно убедиться, на экран будет выведена правильная надпись «не принадлежит».

Рассмотрим условия, записанные в программе. Если не выполняется первое ($x * x + y * y \geq 4$) или второе условие ($x \geq -2$), то на экран не будет выведено никакой надписи, хотя в этом случае программа должна сообщить, что точка не принадлежит указанной области. И только если оба этих условия будут истинными, а условие $y \leq -x$ истинным или ложным, на экран будет выведено какое-то сообщение.

Следовательно, в качестве примеров значений переменных x и y , для которых программа неверно решает поставленную задачу, можно указать следующие:

- 1) $x = 1, y = 1$ — не выполняется условие $x * x + y * y \geq 4$;
- 2) $x = -1, y = 1$ — не выполняется условие $x \geq -2$;
- 3) $x = 2, y = -2$ — не выполняется условие $y \geq 0$.

Для доработки программы необходимо переписать условие. Надпись о принадлежности точки заданной области должна выводиться только при выполнении следующих условий:

- 1) $x * x + y * y \geq 4$;
- 2) $x \geq -2$;
- 3) $y \leq -x$;
- 4) $y \geq 0$.

Надпись же о непринадлежности точки заданной области должна выводиться при ложности хотя бы одного из этих условий. В программу **TestXY** включены все перечисленные условия:

```

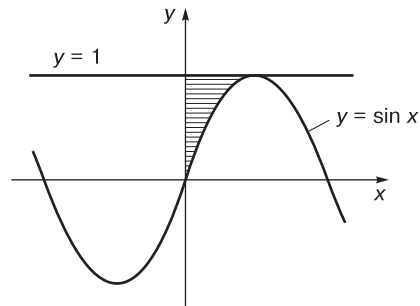
program TestXY;
var
  x, y: single;
begin
  readln(x, y);
  if (x * x + y * y >= 4) and (x >= -2) and (y <= -x)
    and (y >= 0) then

```

```
    write('принадлежит')
  else
    write('не принадлежит');
end.
```

Задача 6. Требовалось написать программу, которая вводит с клавиатуры координаты точки на плоскости (x, y — действительные числа) и определяет принадлежность точки заштрихованной области, включая ее границы. Программист торопился и написал программу неправильно:

```
program TestArea;
var
  x, y: single;
begin
  readln(x, y);
  if y <= 1 then
    if x >= 0 then
      if y >= sin(x) then
        write('принадлежит')
      else
        write('не принадлежит');
    end;
  end.
```



Последовательно выполните следующее:

- 1) приведите пример таких чисел x, y , при которых программа работает неправильно.
- 2) укажите, как нужно доработать программу, чтобы не было случаев ее неправильной работы [6].

Решение.

Проверка принадлежности точки заштрихованной области в программе выполняется с помощью следующих условий:

- 1) $y \leq 1$;
- 2) $y \geq \sin(x)$;
- 3) $x \geq 0$.

Однако в приведенной программе не проверяется условие $x \leq \pi/2$. Поэтому если ввести значение переменной x , большее $\pi/2$, то программа правильно работать не будет.

Рассмотрим условия, записанные в программе. Если не выполняется первое ($y \leq 1$) или второе условие ($x \geq 0$), то на экран не будет выведено никакой надписи, хотя в этом случае должно быть выдано сообщение, что точка не принадлежит указанной области. И только

если условия $y \leq 1$ и $x \geq 0$ истинны, а условие $y \geq \sin(x)$ истинно или ложно, то на экран будет выведено какое-то сообщение.

Следовательно, можно указать следующие значения переменных x и y , для которых программа неверно решает поставленную задачу:

- 1) $y = 2, x = 2$ — не выполняется условие $y \leq 1$;
- 2) $x = -1, y = 2$ — не выполняется условие $x \geq 0$;
- 3) $x = 3, y = 0,5$ — не выполняются условия: $y \geq \sin(x)$ and $x > \pi/2$ and $y \leq 1$.

Для доработки программы необходимо переписать условие анализа исходных данных:

```
program TestArea;
var
  x, y: single;
begin
  readln(x, y);
  if (y <= 1) and (x >= 0) and (y >= sin(x)) and
    (x <= 3.14/2) then
    write('принадлежит')
  else
    write('не принадлежит');
end.
```

Задача 7. Требовалось написать программу, которая решает уравнение $a \cdot |x| = b$ относительно x для любых чисел a и b , введенных с клавиатуры. Все числа считаются действительными. Программист торопился и написал программу неправильно:

```
Program SolveEq;
var
  a, b, x: single;
begin
  readln(a, b, x);
  if a = 0 then
    if b = 0 then
      write('любое число')
    else
      write('нет решений')
  else
    if b = 0 then
      write('x = 0')
```

```
else  
    write('x = ', b / a, ' или x = ', -b / a);  
end.
```

Последовательно выполните три задания:

- 1) приведите пример таких чисел a , b и x , при которых программа неверно решает поставленную задачу;
- 2) укажите, какая часть программы является лишней;
- 3) укажите, как нужно доработать программу, чтобы не было случаев ее неправильной работы [5].

Решение.

Уравнение $a \cdot |x| = b$ имеет два решения: $x = a / b$ и $x = -a / b$. На значения коэффициентов a и b при этом накладываются следующие ограничения: $a > 0$ И $b > 0$ ИЛИ $a < 0$ И $b < 0$, т. е. коэффициенты a и b не могут быть разных знаков.

Учитывая, что в исходной программе отсутствует условие, с помощью которого анализируются знаки переменных a и b , приведем пример значений переменных a , b и x , при которых программа неверно решает поставленную задачу: $a = 2$, $b = -2$, $x = 0$. (Вводимое значение переменной x не влияет на результат решения задачи, однако в тексте задания указано, что требуется привести пример всех трех чисел a , b , x . Поэтому если указать значения только переменных a и b , то это будет рассматриваться как невыполнение части задания.)

Лишней частью в программе является ввод значений переменной x , так как ее значение вычисляется в ходе выполнения программы.

Рассмотрим возможные сочетания значений переменных a и b :

- 1) пусть $a = 0$, тогда если $b = 0$, то решением уравнения является любое число (например: $0 \cdot |x| = 0$, $b = 0$); иначе, если $b \neq 0$, то уравнение решений не имеет (например: $0 \cdot |x| = 2$, $b = 2$);

- 2) пусть $a \neq 0$, тогда:

- если знаки переменных a и b совпадают, то уравнение имеет два решения;
- если знаки переменных a и b не совпадают, то при $b = 0$ уравнение имеет одно решение: $x = 0$, а при $b \neq 0$ уравнение решений не имеет.

Программа будет работать верно, если в нее включены условия, учитывающие **все** приведенные выше сочетания значений переменных a и b (см. программу **SolveEq**):

```
program SolveEq;
var
  a, b, x: single;
begin
  readln(a, b);
  if a = 0 then
    if b = 0 then
      write('любое число')
    else
      write('нет решений')
  else
    if b / a > 0 then
      write('x = ', b / a, ' или x = ', -b / a)
    else
      if b = 0 then
        write('x=0')
      else
        write('нет решений');
end.
```

Задача 8. Требовалось написать программу, которая решает уравнение $a \cdot x + b = 0$ относительно x для любых чисел a и b , введенных с клавиатуры. Все числа считаются действительными. Программист торопился и написал программу неправильно.

```
program SolveEq2;
var
  a, b, x: single;
begin
  readln(a, b, x);
  if b = 0 then
    write('x = 0')
  else
    if a = 0 then
      write('нет решений')
    else
      write('x = ', -b / a);
end.
```

Последовательно выполните три задания:

- 1) приведите пример таких чисел a , b и x , при которых программа неверно решает поставленную задачу;
- 2) укажите, какая часть программы является лишней;

3) укажите, как нужно доработать программу, чтобы не было случаев ее неправильной работы [4].

Решение.

В условиях, записанных в программе, не проверяется возможность одновременного равенства значений переменных a и b нулю, тогда как указанное уравнение при $a = 0$ и $b = 0$ не имеет решения.

Пример значений переменных a , b и x , при которых программа неверно решает поставленную задачу: $a = 0$, $b = 0$, $x = 0$.

Лишней частью в программе является ввод значения переменной x , так как ее значение вычисляется в ходе выполнения программы.

Доработать программу можно, изменив условие анализа значений переменных:

```
program SolveEq2;
var
  a, b, x: single;
begin
  readln(a, b, x);
  if b = 0 then
    if a = 0 then
      write('нет решений')
    else
      write('x = 0')
  else
    if a = 0 then
      write('нет решений')
    else
      write('x = ', -b / a);
end.
```

5.7. Оператор выбора

Оператор выбора позволяет в зависимости от значения <селектора> выполнять определенные действия.

Формат записи оператора выбора:

```
case <селектор> of
  <диапазон 1>: <оператор 1>;
  ...
  <диапазон N>: <оператор N>;
  else <оператор N+1>;
end;
```

где <селектор> — выражение, результат вычисления которого относится к порядковому типу. Значения, указанные в диапазонах, также должны относиться только к порядковому типу. Ветвь **else** является необязательной.

Процесс выполнения оператора выбора:

- 1) вычисляется значение <селектора>;
- 2) проверяется, к какому из возможных диапазонов значений относится вычисленное значение <селектора>. Если такой диапазон найден, то выполняется оператор, связанный с этим диапазоном;
- 3) если значение выражения не принадлежит ни одному из перечисленных диапазонов и присутствует ветвь **else**, то выполняются операторы этой ветви.

Существует несколько способов записи диапазонов значений, которые можно комбинировать:

- 1) перечисление значений, входящих в диапазон, через запятую:

<значение 1>, <значение 2>: <оператор>;

- 2) указание диапазона значений:

<нач. значение> .. <кон. значение>: <оператор>;

Диапазоны значений не должны пересекаться, т. е. одно и то же значение не должно относиться к нескольким диапазонам.

Задача 9. Разработать программу, которая выполняет чтение кода клавиши и выводит информацию о ней.

Решение.

```
program CaseAreaTwo;
var
  ch: char;
begin
  writeln('Нажмите клавишу: ');
  readln(ch);
  case ch of
    '0'..'9':
      writeln('Цифра');
    'a'..'z', 'A'..'Z':
      writeln('Буква');
    '.', ',', ':', ';':
      writeln('Знак препинания');
  else
    writeln('Неизвестная клавиша');
  end;
  readln;
end.
```

Анализ кода нажатой клавиши выполняется с помощью оператора **case**. Для оператора выбора при этом указываются три диапазона, которые включают:

- 1) латинские буквы (строчные и заглавные);
- 2) цифры;
- 3) некоторые знаки препинания.

Если же нажатая клавиша не относится ни к одному из указанных диапазонов, то выполняется ветвь **else**, в которой выводится сообщение о том, что нажатая клавиша неизвестна (см. программу **CaseAreaTwo**).

5.8. Операторы организации циклов

Цикл используется для многократного повторения группы операторов, называемой *телом цикла*. Единичное выполнение тела цикла называется *итерацией*. Каждый цикл должен завершаться после некоторого конечного числа итераций. Условием окончания цикла служит либо заданное число повторений, либо выполнение некоторого условия. Цикл же, который никогда не завершится, называется бесконечным. Как правило, наличие бесконечного цикла является ошибкой, приводящей к «зацикливанию» программы (когда она не может корректно завершить работу).

5.8.1. Оператор цикла **for**

Цикл **for** используется, когда количество итераций цикла заранее известно. В языке Паскаль существует два вида оператора цикла **for**:

- 1) оператор **for-to** — *инкрементный* цикл с параметром;
- 2) оператор **for-downto** — *декрементный* цикл с параметром.

Оператор **for-to** записывается следующим образом:

```
for <сч> := <нз> to <кз> do  
  <оператор>;
```

где <сч> — счетчик (переменная, которая увеличивается на 1 на каждой итерации цикла), <нз> — начальное значение счетчика, <кз> — его конечное значение. В качестве <кз> и <нз> могут использоваться переменные, константы или выражения, относящиеся к эквивалентным целым типам данных.

Цикл **for-to** выполнится хотя бы один раз, если значение $\langle \text{кз} \rangle$ больше или равно значению $\langle \text{нз} \rangle$. Считается, что после окончания работы цикла счетчик принимает неопределенное значение, т. е. заранее неизвестно, чему будет равно его значение.

Задача 10. Разработать программу вычисления факториала заданного числа.

Решение.

Факториалом числа n (обозначается как $n!$) называется произведение всех натуральных чисел до n включительно:

$$n! = 1 \cdot 2 \cdot \dots \cdot n = \prod_{i=1}^n i.$$

Факториал нуля при этом равен единице ($0! = 1$). Факториал определен только для целых неотрицательных чисел.

В программе значение факториала можно вычислить следующим образом:

$$n_i = i \cdot n_{i-1},$$

где n_i — значение факториала n на текущем (i -м) шаге, а n_{i-1} — значение факториала n на предыдущем шаге. То есть значение факториала на i -м шаге выполнения цикла рассчитывается на основе значения факториала, полученного на $(i - 1)$ -м шаге (используется *рекуррентная формула*).

```

program ForFactorial;
var
    n, i, f: longint;
begin
    write('Введите n (n>=2) = ');
    readln(n);
    f := 1;
    for i := 2 to n do
        f := f*i;
    writeln(n, '!= ', f);
    readln;
end.

```

В следующей таблице отражен процесс вычисления факториала для $n = 4$:

i	$f := f \cdot i$	f
2	$f := 1 \cdot 2 = 2$	2
3	$f := 2 \cdot 3 = 6$	6
4	$f := 4 \cdot 6 = 24$	24

Оператор цикла **for-downto** записывается следующим образом:

```
for <сч> := <нз> downto <кз> do  
  <оператор>;
```

где <сч> — счетчик (переменная, которая уменьшается на 1 на каждой итерации цикла), <нз> — начальное значение счетчика, <кз> — конечное значение.

Такой цикл работает аналогично циклу **for-to**, но, в отличие от него, значение счетчика на каждой итерации здесь уменьшается на 1. Соответственно, для выполнения такого цикла необходимо, чтобы соблюдалось условие: $\langle \text{нз} \rangle > \langle \text{кз} \rangle$.

5.8.2. Оператор цикла **while**

Оператор цикла **while** записывается следующим образом:

```
while <условие> do  
  <оператор>;
```

Выполнение оператора начинается с вычисления значения выражения, записанного в <условии>. Если значение этого выражения равно **true**, то выполняется <оператор>; иначе происходит выход из цикла — переход к оператору, следующему за телом цикла. Таким образом, цикл выполняется до тех пор, пока результат вычисления логического выражения в <условии> — истина.

Если при первом вычислении значения <условия> выяснится, что оно ложно, то цикл не выполнится совсем, а управление будет сразу передано следующему за циклом оператору.

Пример записи бесконечного цикла с помощью оператора **while** показан в программе **IfiniteExample**.

```
program IfiniteExample;  
begin  
  while (true) do  
    writeln('Бесконечный цикл');  
end.
```

Задача 11. Даны целые положительные числа N и K . Используя только операции сложения и вычитания, найти частное от деления нацело N на K , а также остаток от деления.

Решение.

Операцию деления можно заменить многократным использованием операции вычитания. Поэтому при определении частного

и остатка для повторения операции вычитания в программе **PrvRem** используется оператор цикла **while**, а также следующие переменные:

- 1) n — делимое (изменяется в ходе выполнения программы);
- 2) k — делитель;
- 3) rem — остаток от деления N на K ;
- 4) $priv$ — частное от деления N на K ;
- 5) n_tmp — делимое (копия переменной n).

```

program PrvRem;
var
  n, k, n_tmp, rem, priv: word;
begin
  write('Введите делимое = ');
  readln(n);
  write('Введите делитель = ');
  readln(k);
  n_tmp := n; x := n;
  priv := 0; rem := 0;
  while n > k do
    begin
      n := n - k;
      priv := priv + 1;
      rem := rem + k;
    end;
    if (n_tmp - rem) > 0 then
      rem := n_tmp - rem
    else
      rem := 0;
    writeln('Частное = ', priv);
    writeln('Остаток = ', rem);
    readln;
  end.

```

Цикл выполняется до тех пор, пока делимое больше делителя. При этом на каждом шаге цикла производятся следующие действия:

- 1) делимое уменьшается на величину делителя ($n := n - k$);
- 2) частное увеличивается на единицу ($priv := priv + 1$);
- 3) увеличивается значение переменной rem ($rem := rem + k$).

После окончания работы цикла вычисляется значение остатка.

Заметим, что в теле цикла находятся три оператора, поэтому они объединяются с помощью операторных скобок **begin .. end**. Если

же операторные скобки не использовать, то к циклу **while** будет отнесен только первый оператор: $x := n - k$, что приведет к неверной работе программы.

Задача 12. В результате выполнения фрагмента алгоритма:

```
while n <> 0 do
begin
  write (2 * (n mod 10));
  n := n div 10;
end;
```

было напечатано число 104128. Предполагается, что перед выполнением этого фрагмента алгоритма значение переменной n было равно 120814. Верно ли данное утверждение?

Решение.

Общие замечания:

1) div — операция деления нацело; запись $n \text{ div } 10$ означает, что у числа n отбрасывается последняя цифра (например, $321 \text{ div } 10 = 32$);

2) mod — операция вычисления остатка от деления; запись $n \text{ mod } 10$ означает выделение последней цифры числа (например, $321 \text{ mod } 10 = 1$).

Рассмотрим теперь процесс выполнения приведенного фрагмента программы для числа 821401 по шагам:

1) $n = 120814$;

1.1) условие $n \neq 0$ истинно; $n \text{ mod } 10 = 4$; $2 \cdot 4 = 8$, — выводится 8;

1.2) $n := 120814 \text{ div } 10 = 12081$;

2) $n = 12081$;

2.1) условие $n \neq 0$ истинно; $n \text{ mod } 10 = 1$; $2 \cdot 1 = 2$, — выводится 2;

2.2) $n := 12081 \text{ div } 10 = 1208$;

3) $n = 1208$;

3.1) условие $n \neq 0$ истинно; $n \text{ mod } 10 = 8$; $2 \cdot 8 = 16$, — выводится 16;

3.2) $n := 1208 \text{ div } 10 = 120$;

4) $n = 120$;

4.1) условие $n \neq 0$ истинно; $n \text{ mod } 10 = 0$; $2 \cdot 0 = 0$, — выводится 0;

4.2) $n := 120 \text{ div } 10 = 12$;

5) $n = 12$;

5.1) условие $n \neq 0$ истинно; $n \text{ mod } 10 = 2$; $2 \cdot 2 = 4$, выводится 4;

5.2) $n := 12 \text{ div } 10 = 1$;

6) $n = 1$

6.1) условие $n \neq 0$ истинно; $n \bmod 10 = 1$; $2 \cdot 1 = 2$, выводится 2;

6.2) $n := 1 \operatorname{div} 10 = 0$;

7) $n = 0$;

7.1) условие $n \neq 0$ ложно; программа прекращает работу.

Таким образом, в результате выполнения этой программы на экране появится строка 8216042 (хотя требовалось 104128). Данная строка не эквивалентна строке, приведенной в тексте задания, следовательно, значение переменной n до выполнения алгоритма не было равно 8216042. (Уже на первых шагах выполнения программы было очевидно, что получаемая строка не соответствует искомой: несовпадения начались с самой первой цифры. Поэтому после того, как получено первое несоответствие, можно прекратить анализ числа.)

Ответ: приведенное утверждение ложно.

5.8.3. Оператор цикла **repeat**

Оператор цикла **repeat** записывается следующим образом:

```
repeat <операторы>  
until <условие>;
```

Порядок работы цикла **repeat**:

- 1) выполняются <операторы>, находящиеся в теле цикла;
- 2) вычисляется значение <условие>: если оно *ложно*, то выполняется тело цикла, а если <условие> истинно, то оператор **repeat** прекращает свою работу и управление передается оператору, который следует за циклом.

Особенность оператора **repeat** состоит в том, что тело цикла всегда выполняется хотя бы один раз, даже если логическое выражение изначально является истинным, так как значение <условие> вычисляется после первого выполнения цикла.

Отметим, что зарезервированные слова **repeat** и **until** в данном случае сами служат операторными скобками, поэтому если в теле цикла должно выполняться несколько операторов, то использовать операторные скобки **begin** .. **end** уже нет необходимости.

Задача 13. Дано целое число n ($n > 0$), являющееся некоторой степенью числа 2: $n = 2^k$. Найти целое число k — показатель этой степени.

Решение.

Определение показателя степени (k) основано на последовательном делении исходного числа (n) на основание степени, т. е. на 2 (см. программу **RepeatExample**).

```
program RepeatExample;
var
  n: single;
  k: byte;
begin
  readln(n);
  k := 0;
  repeat
    n := n / 2;
    k := k + 1;
  until (n < 2);
  writeln(k);
end.
```

Для решения поставленной задачи в программе используется оператор цикла **repeat**. При этом на каждой итерации цикла выполняются следующие действия:

- 1) исходное число делится на основание степени: $n := n / 2$;
- 2) подсчитывается количество выполненных операций деления: $k := k + 1$.

Цикл должен выполняться до тех пор, пока значение переменной n больше или равно основанию степени. После окончания работы цикла переменная k будет содержать искомый показатель степени.

5.8.4. Вложенные операторы циклов

По отношению друг к другу циклы могут быть внешними и внутренними (вложенными). *Внешним* называется цикл, полностью содержащий в себе другой цикл. Тогда второй цикл по отношению к первому является *внутренним*. Необходимо отметить, что при вложении циклов **for** в качестве счетчиков нужно использовать переменные с разными идентификаторами.

Задача 14. Найти все трехзначные числа, у которых первая и вторая, а также вторая и третья цифры различны.

Решение.

Для перебора трехзначных чисел можно использовать три вложенных цикла **for**. Каждый из этих циклов используется для перебора цифр одного из трех разрядов числа (первый — для сотен, второй — для десятков, третий — для единиц). При этом переменная-счетчик первого цикла должна изменяться от 1 до 9, так как у трехзначных чисел первая цифра обязательно должна быть ненулевой. Решение задачи приведено в программе **DiffDigits**.

```
program DiffDigits;
var
  d1, d2, d3: byte;
begin
  for d1 := 1 to 9 do
    for d2 := 0 to 9 do
      for d3 := 0 to 9 do
        if (d1<>d2) and (d2<>d3) then
          writeln(d1, d2, d3);
      readln;
    end.
```

5.8.5. Операторы **break** и **continue**

Операторы **break** и **continue** используются для управления выполнением цикла. Оператор **break** досрочно прерывает работу цикла и передает управление на следующий за ним оператор. Оператор **continue** прерывает работу только текущей итерации цикла и передает управление следующей итерации (цикл **repeat-until**) или на предшествующую ей проверку (циклы **for-to**, **for-downto**, **while**). Принципы работы указанных операторов демонстрируют программы **BreakExample** и **ContinueExample**.

```
program BreakExample;
var
  num: integer;
begin
  repeat
    write('Введите число: ');
    readln(num);
    if num > 1 then
```

```
        writeln(num)
    else
        break;
    until (false);
end.
```

Программа **BreakExample** выполняется следующим образом:

- 1) вводится значение переменной *num*;
- 2) если введенное значение больше единицы, то это значение выводится на экран и выполняется переход на шаг 1; иначе выполняется переход на шаг 3;
- 3) производится выход из цикла с помощью оператора **break**.

```
program ContinueExample;
var
    i: integer;
begin
    for i := 1 to 10 do
    begin
        if i >= 5 then
            continue;
        writeln(i);
    end;
    readln;
end.
```

В программе `ContinueExample` организован цикл от 1 до 10. Пока значение переменной-счетчика меньше или равно 5, на экран выводится значение счетчика. Как только это значение станет больше 5, выполняется оператор **continue**, который прерывает процесс выполнения текущей итерации цикла и передает управление следующей итерации. В результате при $i \geq 5$ значение переменной-счетчика на экран выводиться не будет.

Операторы **break** и **continue** взаимодействуют только с тем циклом, в котором они вызываются, т. е. при использовании двух циклов, один из которых вложен в другой, указанные операторы могут прервать только один из этих циклов, но не оба сразу.

Задача 15. Запишите значение переменной *b* после выполнения фрагмента блок-схемы алгоритма [7]:

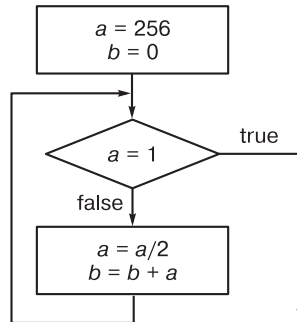


Рис. 5.4. Блок-схема алгоритма

Решение.

Блок-схема, показанная на рис. 5.4, описывает алгоритм, в который входит цикл. До начала выполнения этого цикла устанавливаются значения переменных a и b , а затем в цикле вычисляются их новые значения.

Запишем процесс выполнения алгоритма в виде таблицы:

Шаг	Операторы
0	$a := 256; b := 0;$
1	$a = 1$ – ложь ($256 \neq 1$)
2	$a := 256/2, a = 128$ $b := 0 + 128, b = 128$
3	$a = 1$ – ложь ($128 \neq 1$)
4	$a := 128/2, a = 64$ $b := 64 + 128, b = 192$
5	$a = 1$ – ложь ($64 \neq 1$)
6	$a := 64/2, a = 32$ $b := 192 + 32, b = 224$
7	$a = 1$ – ложь ($32 \neq 1$)
8	$a := 32/2, a = 16$ $b := 224 + 16, b = 240$
9	$a = 1$ – ложь ($16 \neq 1$)
10	$a := 16/2, a = 8$ $b := 240 + 8, b = 248$

Шаг	Операторы
11	$a = 1 - \text{ложь} \quad (8 = 1)$
12	$a := 8/2, a = 4$ $b := 248 + 4, b = 252$
13	$a = 1 - \text{ложь} \quad (4 = 1)$
14	$a := 4/2, a = 2$ $b := 252 + 2, b = 254$
15	$a = 1 - \text{ложь} \quad (2 = 1)$
16	$a := 2/2, a = 1$ $b := 254 + 1, b = 255$
17	$a = 1 - \text{истина} \quad (1 = 1)$

Таким образом, после выполнения фрагмента алгоритма значение переменной b будет равно 255.

Ответ: 255.

Задача 16. Определите значение переменной m после выполнения блок-схемы алгоритма [5]:

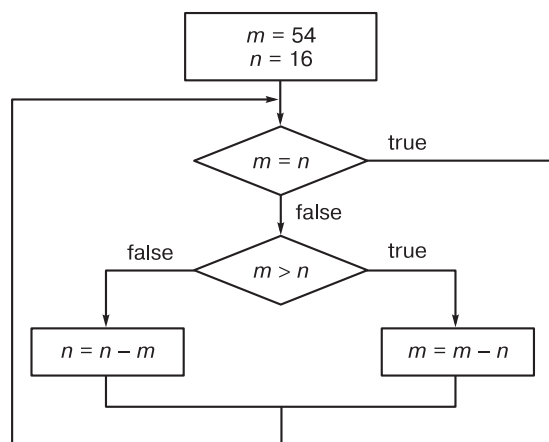


Рис. 5.5. Блок-схема алгоритма

Решение.

Алгоритм, описываемый этой блок-схемой, выполняется следующим образом:

- 1) устанавливаются начальные значения переменных m и n ;
- 2) в цикле вычисляются значения переменных m и n .

Запишем процесс выполнения алгоритма в виде таблицы:

Шаг	Операторы
0	$m = 54, n = 16$
1	$m = n - \text{ложь } (54 = 16),$ $m > n - \text{истина } (54 > 16)$
2	$m := 54 - 16, m = 38$ $m = 38, n = 16$
3	$m = n - \text{ложь } (38 = 16),$ $m > n - \text{истина } (38 > 16)$
4	$m := 38 - 16, m = 22$ $m = 22, n = 16$
5	$m = n - \text{ложь } (22 = 16),$ $m > n - \text{истина } (22 > 16)$
6	$m := 22 - 16, m = 6$ $m = 6, n = 16$
7	$m = n - \text{ложь } (6 = 16),$ $m > n - \text{ложь } (6 > 16)$
8	$n := 16 - 6, n = 10$ $m = 6, n = 10$
9	$m = n - \text{ложь } (6 = 10),$ $m > n - \text{ложь } (6 > 10)$
10	$n := 10 - 6, n = 4$ $m = 6, n = 4$
11	$m = n - \text{ложь } (6 = 4),$ $m > n - \text{истина } (6 > 4)$
12	$m := 6 - 4, m = 2$ $m = 2, n = 4$
13	$m = n - \text{ложь } (2 = 4),$ $m > n - \text{ложь } (2 > 4)$
14	$n := 4 - 2, n = 2$ $m = 2, n = 2$
15	$m = n - \text{истина } (2 = 2)$

Таким образом, после выполнения фрагмента алгоритма значение переменной m будет равно 2.

Ответ: 2.

5.9. Работа с массивами

5.9.1. Определение массива

Массив — это объект, образованный из элементов одного типа. Массив определяется следующим образом:

```
var
    <имя массива>: array [<ДИ1>, ..., <ДИN>] of
                                <тип элементов массива>;
```

где <ДИ> — диапазон индексов (<ДИ> = <Ни .. Ки>, где Ни — индекс первого элемента массива, Ки — индекс последнего элемента массива), а N — количество диапазонов индексов. На тип данных элементов массива ограничений не накладывается, кроме одного: невозможен массив файлов.

С каждым элементом массива связано уникальное число, называемое *индексом элемента*.

Количество диапазонов индексов определяет *размерность массива*:

- один диапазон — одномерный массив;
- два диапазона — двумерный массив и т. д.

Одномерный массив можно представить в виде таблицы, состоящей из одной строки и нескольких столбцов, где в первой строке таблицы показаны номера элементов массива, во второй — значения элементов.

Индекс элемента →	1	2	3	4	5
Значение элемента →	10	200	1	0	4

Пример объявления одномерного массива из 5 элементов типа `byte`:

```
var
    a: array [1..5] of byte;
```

Двумерный массив можно представить как таблицу:

	1	2
1	0	5
2	7	120
3	240	1

Пример объявления двумерного массива размерами 3×2 :

```
var
    b: array [1..3, 1..2] of byte;
```

Здесь первый диапазон индексов определяет возможные значения номеров строк, а второй — значения номеров столбцов.

Обращение к элементу массива выполняется указанием имени массива и значения индекса (номера) его элемента в квадратных скобках. Например, в программе **ArrayIndex** первому элементу массива *a* присваивается значение 10:

```
program ArrayIndex;
var
  a: array [1..10] of byte;
begin
  a[1] := 10;
end.
```

При работе с массивами удобно описывать конечные значения диапазонов индексов в виде констант, а затем использовать их в тексте программы, например:

```
program ArrayAndConst;
const
  n = 10;
var
  a: array [1..n] of byte;
  i, sum: integer;
begin
  for i := 1 to n do
    a[i] := random(5);
  sum := 0;
  for i := 1 to n do
    sum := sum + a[i];
  writeln('Сумма элементов массива = ', sum);
  readln;
end.
```

Если теперь потребуется изменить значение конечного индекса некоторого диапазона, то достаточно изменить значение константы, а не разыскивать и менять значения индексов во всей программе.

Определение (описание) типа массива выглядит следующим образом:

```
type
  <имя массива> = array [<ДИ1>, ..., <ДИN>] of
    <тип элементов массива>;
```

После такого описания тип можно использовать при объявлении соответствующих переменных, например:


```
type
  arr_type = array [1..10] of byte;
var
  arr1, arr2, arr3: arr_type;
```

Алгоритмы обработки значений элементов массива обычно основаны на использовании операторов цикла.

5.9.2. Инициализация массива

Существует несколько способов заполнения массива значениями.

Использование оператора присваивания

Пример использования оператора присваивания для записи в элементы массива некоторых значений показан в программе **ArrayInit**.

```
program ArrayInit;
var
  a: array [1..3] of byte;
begin
  a[1] := 1; // заполнение 1-го элемента
  a[2] := 4; // заполнение 2-го элемента
  a[3] := 7; // заполнение 3-го элемента
end.
```

Заполнение массива при объявлении

Инициализировать массив при его объявлении можно в разделе **var** или в разделе **const**. Для одномерного массива начальные значения элементов задаются через запятую и заключаются в круглые скобки:

```
type
  mass = array [1..3] of byte; {объявление типа}
const
  a: array [-1..1] of byte = (0,0,0);
  b: mass = (1, 2, 3);
var
  c: array [-2..0] of byte = (1, 2, 3);
```

Двумерный массив можно рассматривать как одномерный массив, состоящий из элементов — одномерных массивов:

```
type
  mass = array [1..3, 1..2] of byte;
```

```

const
  a: array [-1..1] of byte = (0,0,0); {одномерный}
  b: mass = ((1,2),(3,4),(5,6)); {двумерный}
  ss: array [0..1] of char = ('s', 'v');
var
  c: array [-2..0, 1..2] of byte =
      ((1, 2), (3, 4), (5, 6));

```

Изменить значения элементов массива *a*, который заполнен в разделе **const**, *нельзя*. Для обхода данного ограничения можно сделать следующее:

- 1) объявить в разделе **var** массив *aNew* точно такого же размера и типа, как и *a*;
- 2) инициализировать массив *aNew* в программе, используя массив *a*.

После выполнения указанных действий с массивом *aNew* можно осуществлять все допустимые действия, в том числе изменение значений его элементов.

Для инициализации массива *aNew* можно использовать или поэлементную передачу данных из массива *a* в *aNew* в цикле (см. программу **ArrayInitExample2**), или оператор присваивания (см. программу **ArrayInitExample1**).

```

program ArrayInitExample1;
const
  a: array [-1..1] of byte = (1,2,3);
var
  i: integer;
  aNew: array [-1..1] of byte;
begin
  a_new := a;
  for i:= -1 to 1 do
    writeln('a_new['i,']= ', a_new[i]);
  readln;
end.

program ArrayInitExample2;
const
  a: array [-1..1] of byte = (1,2,3);
var
  i: integer;
  aNew: array [-1..1] of byte;

```

```
begin
  for i:= -1 to 1 do
    a_new[i] := a[i];
  for i:= -1 to 1 do
    writeln('a_new[' , i, ']=', a_new[i]);
  readln;
end.
```

Инициализация массива случайными числами

Заполнение массива случайными числами основано на использовании функции **random**. Для генерации случайного числа на полуинтервале $[a; b)$ можно использовать следующее выражение:

```
r := a + (b-a)*random;
```

где r — вещественная переменная, в которую записывается случайное число ($r \in [a; b)$). В программе **FillRandom1d** показан пример заполнения случайными числами одномерного массива, а в программе **FillRandom2d** — двумерного.

```
program FillRandom1d;
const
  n = 5;
var
  a: array [1..n] of single;
  i, x, y: byte;
begin
  x := 10;
  y := 20;
  for i := 1 to n do
    a[i] := x + (y-x)*random;
  end.

program FillRandom2d;
const
  n = 10;
  m = 9;
var
  i, j, x, y: byte;
  a: array [1..n, 1..m] of single;
begin
  x := 10;
  y := 20;
```

```
    for i := 1 to n do
      for j := 1 to m do
        a[i, j] := x + (y-x)*random;
      end.
    end.
```

Вызов функции `radomize` позволяет при каждом запуске программы генерировать новый набор случайных чисел, используемых при заполнении массива.

Ввод значений элементов массива с клавиатуры

Ввод значений элементов массива с клавиатуры основан на использовании процедуры `readln` и оператора цикла (обычно — `for`). Программы `Input1dArray` и `Input2dArray` демонстрируют ввод значений одномерного и двумерного массивов.

```
program Input1dArray;
const
  n = 5;
var
  a: array [1..n] of byte;
  i: integer;
begin
  for i := 1 to n do
    begin
      write('Вв. a[' , i, ']=');
      readln(a[i]);
    end;
  end.

program Input2dArray;
const
  n = 2;
  m = 3;
var
  i, j: byte;
  a: array [1..n, 1..m] of byte;
begin
  for i := 1 to n do
    for j := 1 to m do
      begin
        write('Вв. a[' , i, ']' , '[' , j, ']=');
```

```
        read(a[i, j]);  
    end;  
    readln;  
end.
```

5.9.3. Вывод массива

Вывод одномерного массива

Для вывода элементов одномерного массива используется оператор цикла и процедура вывода данных `write` (см. программу **Out1dArray**). В аргументах команды `write(a[i]:5:2, ' ');` пробел нужен, чтобы отделять выводимые числа (элементы массива) друг от друга.

```
program Out1dArray;  
const  
    n = 5;  
var  
    a: array [1..n] of single;  
    i: integer;  
begin  
    randomize;  
    for i := 1 to n do  
        a[i] := random(100)/10;  
    for i := 1 to n do  
        write(a[i]:5:2, ' ');  
    readln;  
end.
```

Результат вывода одномерного массива:

```
5.40  5.90  7.10  8.40  6.00
```

Элементы одномерного массива можно вывести в столбик. Для этого нужно заменить команду `write(a[i]:5:2, ' ');` на `writeln(a[i]:5:2).`

Вывод двумерного массива

Программа **Out2dArray** выводит на экран двумерный массив в виде таблицы. Для этого используется два цикла, один из которых (цикл по *j*) вложен в другой (в цикл по *i*).

```
program Out2dArray;
const
  n = 5;
  m = 7;
var
  a: array [1..n, 1..m] of single;
  i, j: integer;
begin
  for i := 1 to n do
    for j := 1 to m do
      a[i, j] := random(100) / 10;
  for i := 1 to n do
    begin
      for j := 1 to m do
        write(a[i, j]:5:2, ' ');
      writeln;
    end;
  readln;
end.
```

Данная программа работает следующим образом:

- 1) вывод элементов j -й строки в цикле по j ;
- 2) перевод строки с помощью процедуры `writeln`.

Указанные шаги выполняются до тех пор, пока не будут выведены все строки массива в цикле по i .

Результат работы программы:

5.40	5.90	7.10	8.40	6.00	8.50	5.40
8.40	4.20	6.20	6.40	3.80	4.30	2.90
8.90	0.50	9.60	2.70	3.80	4.70	7.90
8.10	5.20	4.70	5.60	3.90	9.20	8.30
0.70	3.30	0.80	6.40	0.20	3.60	8.30

5.9.4 Сортировка одномерного массива

Сортировкой, или упорядочением, массива называется расположение его элементов по возрастанию или по убыванию.

Метод «пузырька»

Здесь на каждом шаге выполняются следующие действия (рассматривается сортировка по возрастанию):

- 1) сравнивается пара соседних элементов массива: i -й и $(i + 1)$ -й;
- 2) если i -й элемент больше $(i + 1)$ -го, то они меняются местами;

3) первый и второй шаги выполняются для всех соседних элементов массива.

Таким образом, после выполнения одного прохода по массиву на последнее место в нем помещается минимальный элемент, после чего указанные действия повторяются для оставшихся элементов.

Заметим, что на втором и последующих шагах уже нет необходимости рассматривать ранее перемещенные в конец массива элементы, так как они заведомо больше оставшихся. Другими словами, на i -м шаге не проверяются элементы, стоящие на позициях с номерами больше i (т. е. элементы, индекс которых превышает i).

Программа **SortArray** выполняет упорядочение массива *arr* по возрастанию.

```
program SortArray;
const
  N = 5;
var
  i, j, t: byte;
  arr: array [1..n] of byte;
begin
  arr[1] := 7; arr[2] := 2;
  arr[3] := 5; arr[5] := 1;
  arr[4] := 3;
  writeln('Исходный массив: ');
  for i := 1 to N do
    write(arr[i], ' ');
  for i := 1 to N-1 do
    for j := 1 to N-i do
      if arr[j] > arr[j+1] then
        begin
          t := arr[j];
          arr[j] := arr[j+1];
          arr[j+1] := t;
        end;
    writeln;
  writeln('Отсорт. массив: ');
  for i:=1 to N do
    write(arr[i], ' ');
  readln;
end.
```

В нижеследующей таблице приведен процесс перемещения первого (максимального) элемента массива:

№ элемента	0	$i = 1, j = 1$	$i = 1, j = 2$	$i = 1, j = 3$	$i = 1, j = 4$
1	3	3	3	3	3
2	7	7	1	1	1
3	1	1	7	5	5
4	5	5	5	7	2
5	2	2	2	2	7

Если же требуется отсортировать массив по убыванию, то следует поменять знак «>» в выражении **if** `arr[j] > arr[j+1]` **then** на противоположный.

В описанном алгоритме сортировки используется прием, с помощью которого две переменные обмениваются значениями. Данное действие выполняется с помощью дополнительной переменной *t* (см. программу **ChangeVars**).

```

program ChangeVars;
var
    a, b, t: byte;
begin
    a := 2;
    b := 10;
    writeln('a = ', a, ', b = ', b);
    t := a;
    a := b;
    b := t;
    writeln('a = ', a, ', b = ', b);
    readln;
end.

```

Сортировка выбором

Принцип работы алгоритма состоит в следующем:

- 1) $i = 1$;
- 2) если $i \leq N$, то надо выбрать минимальный элемент *min* среди элементов $A[i] \dots A[N]$, иначе работа алгоритма завершается;
- 3) найденный минимальный элемент *min* обменивается местами с *i*-м элементом массива;
- 4) $i = i + 1$ и переход на второй шаг.

Программная реализация:

```

program SelectionSort;
const
  N = 10;
var
  i, j, k, x: integer;
  A: array[1.. N] of integer;
begin
  for i := 1 to N do
    A[i] := Random(20);
  for i := 1 to N do
    write(' ', A[i]);
  writeln;
  for i := 1 to N do
  begin
    k := i;
    x := A[i];
    for j := i to N do
      if A[j] < x then
      begin
        k := j;
        x := A[j];
      end;
    A[k] := A[i];
    A[i] := x;
  end;
  for i := 1 to N do
    write(' ', A[i]);
  writeln;
end.

```

Пример работы алгоритма:

Исходный массив A:

1	2	3	4	5	6	7
3	-4	2	0	-1	5	7

$i = 1, \min(A[1] \dots A[7]) = -4:$

-4	3	2	0	-1	5	7
----	---	---	---	----	---	---

$i = 2, \min(A[2] \dots A[7]) = -1:$

-4	-1	2	0	3	5	7
----	----	---	---	---	---	---

...

Сортировка вставками

Пусть A — исходный массив. Рассмотрим алгоритм сортировки вставками:

- 1) создаем массив B , количество элементов которого равно N ;
- 2) если $i \leq N$, то выбираем i -й элемент исходного массива ($A[i]$), иначе работа алгоритма завершается;
- 3) выделяем в конце массива B свободную ячейку;
- 4) анализируем j -й элемент массива B , стоящий перед пустой ячейкой:
 - 4.1) если $B[j] > A[i]$, то перемещаем элемент $B[j]$ в свободную ячейку;
 - 4.2) если $B[j] \leq A[i]$ или пустая ячейка стоит в начале массива, то помещаем вставляемый элемент в пустую ячейку;
- 5) $i = i + 1$ и переходим на шаг 2.

Программная реализация:

```
program InsertionSort;
const
  N = 10;
var
  A, B: array[1.. N] of integer;
  i, j: integer;
begin
  for i := 1 to N do
    A[i] := Random(10) - 5;
  for i := 1 to N do
    begin
      j := i;
      while (j > 1) and (B[j-1] > A[i]) do
        begin
          B[j] := B[j-1];
          j := j - 1;
        end;
      B[j] := A[i];
    end;
  for i := 1 to N do
    write(' ', A[i]);
  writeln;
  for i := 1 to N do
    write(' ', B[i]);
end.
```

Пример работы алгоритма:

Исходный массив A :

3	-4	2	0	-1	5	7
---	----	---	---	----	---	---

Массив B после выполнения трех шагов:

-4	2	3
----	---	---

Вставка элемента с индексом 4, т. е. значения 0:

$3 > 0$ — истина:

-4	2	0	3
----	---	---	---

$2 > 0$ — истина:

-4	0	2	3
----	---	---	---

$-4 > 0$ — ложь. Результат вставки:

-4	0	2	3
----	---	---	---

...

Быстрая сортировка

Алгоритм быстрой сортировки:

- 1) выбрать в массиве базовый элемент (в качестве него можно взять элемент в середине массива);
- 2) разделить массив — все элементы, меньшие или равные базовому элементу, поместить слева от него, а все элементы, большие базового, — справа от него:
 - 2.1) приравнять индексы i и j индексам элементов, стоящих слева и справа от базового;
 - 2.2) пока $i < j$, последовательно увеличивать индекс i до тех пор, пока i -й элемент не станет больше базового;
 - 2.3) пока $i < j$, последовательно уменьшать индекс j до тех пор, пока j -й элемент не станет меньше или равен базовому;
 - 2.4) если $i < j$, то обменять местами элементы с индексами i и j и продолжить операцию деления массива с тех значений i и j , которые были достигнуты до этого;
- 3) рекурсивно упорядочить массивы, лежащие слева и справа от базового элемента.

Программная реализация:

```

program QuickSort;
const
  N = 7;
var
  A: array[1..N] of integer;
  
```

```
    i, j, T: integer;
procedure Sort(p, q: integer);
var
    i, j, r: integer;
begin
    if p < q then
    begin
        r := A[p];
        i := p - 1;
        j := q + 1;
        while i < j do
        begin
            repeat
                i := i + 1;
            until A[i] >= r;
            repeat
                j := j - 1;
            until A[j] <= r;
            if i < j then
            begin
                T := A[i];
                A[i] := A[j];
                A[j] := T;
            end;
        end;
        Sort(p, j);
        Sort(j + 1, q);
    end;
end;
begin
    for i := 1 to N do
        A[i] := Random(20) - 10;
    for i := 1 to N do
        write(' ', A[i]);
    writeln;
    Sort(1, N);
    for i := 1 to N do
        write(' ', A[i]);
    writeln;
end.
```

Сортировка слиянием

Алгоритм сортировки слиянием заключается в выполнении следующих шагов:

- 1) сортируемый массив разбивается на две части примерно одинакового размера;
- 2) каждая из полученных частей сортируется отдельно, т. е. для каждой части массива выполняется первый шаг. Рекурсивное разбиение массива на подмассивы производится до тех пор, пока размер массива не станет равен единице (поскольку любой массив длины 1 можно считать упорядоченным);
- 3) два уже упорядоченных массива половинного размера соединяются в один.

Объединение двух отсортированных массивов A и B выполняется следующим образом: сравниваются два первых элемента $A[i]$ и $B[j]$, выбирается меньший элемент, который записывается в результирующий массив C , а после того, как все элементы одного из массивов перемещены в массив C , оставшиеся элементы другого массива добавляются в результирующий массив.

Программная реализация:

```
program JoinSort;
const
  N = 4;
var
  A, B: array[1..N] of integer;
procedure Sort(p, q: integer);
var
  r, i, j, k: integer;
begin
  if p < q then
  begin
    Sort(p, (p + q) div 2);
    Sort((p + q) div 2 + 1, q);
    r := (p + q) div 2;
    i := p;
    j := r + 1;
    for k := p to q do
      if (i <= r) and ((j > q) or (a[i] < a[j])) then
      begin
        b[k] := a[i];
        i := i + 1;
      end
    end
  end
end;
```

```

        end
        else
        begin
            b[k] := a[j];
            j := j + 1;
        end;
        for k := p to q do
            a[k] := b[k];
        end;
    end;
end;
var
    i: integer;
begin
    for i := 1 to N do
        A[i] := Random(20);
    end;
    for i := 1 to N do
        write(' ', A[i]);
    end;
    writeln;
    Sort(1, N);
    for i := 1 to N do
        write(' ', A[i]);
    end;
    writeln;
end.

```

Сортировка методом Шейкера

Алгоритм Шейкера состоит из следующих шагов:

- 1) $i = 1$;
- 2) если $i \leq (N \div 2) + 1$, то выбрать минимальный и максимальный элементы среди элементов $A[i] \dots A[N - i + 1]$, иначе работа алгоритма завершается;
- 3) обменять местами минимальный элемент min и элемент $A[i]$; обменять местами максимальный элемент max и элемент $A[N - i + 1]$;
- 4) $i = i + 1$ и переход на шаг 2.

Программная реализация:

```

program ShakerSort;
const
    N = 7;
var
    A: array[1..N] of integer;

```

```

    i, j, p, st, ed, k: integer;
    min, max: integer;
begin
    for i := 1 to N do
        A[i] := Random(20);
    for i := 1 to N do
        write(' ', A[i]);
    writeln;
    for i := 1 to (N div 2) + 1 do
    begin
        st := i;
        ed := N - i + 1;
        min := st;
        max := ed;
        for k := st to ed do
        begin
            if A[k] > A[max] then
                max := k;
            if A[k] < A[min] then
                min := k;
        end;
        p := A[min];
        A[min] := A[st];
        A[st] := p;
        if (max = st) then
            max := min;
        p := A[max];
        A[max] := A[ed];
        A[ed] := p;
    end;
    for i := 1 to N do
        write(' ', A[i]);
    writeln;
end.

```

Пример работы алгоритма:

Исходный массив A :

3	-4	2	0	7	-1	5
---	----	---	---	---	----	---

$i = 1, N - i + 1 = 7, min = -4, max = 7$:

-4	3	2	0	5	-1	7
----	---	---	---	---	----	---

$i = 2, N - 2 + 1 = 6, \min = -1, \max = 5$:

-4	-1	2	0	3	5	7
----	----	---	---	---	---	---

...

Сортировка Шелла

Алгоритм Шелла:

- 1) $d = N \div 2$;
- 2) если $d > 0$, то сформировать группы элементов массива, составленных из элементов, находящихся на расстоянии $d/2$ друг от друга, иначе работы алгоритма завершается;
- 3) выполнить сортировку элементов, входящих в каждую группу;
- 4) $d = d \div 2$ и переход на шаг 2.

Рассмотрим пример сортировки массива при $N = 8$:

- 1) первый шаг: $d = 8 \div 2 \Rightarrow d = 4$; формируются группы из элементов с индексами (1, 5), (2, 6), (3, 7), (4, 8); после формирования групп сортируются элементы, входящие в каждую группу;
- 2) второй шаг: $d = 4 \div 2 \Rightarrow d = 2$; формируются две группы элементов с индексами (1, 3, 5, 7) и (2, 4, 6, 8); выполняется сортировка элементов внутри каждой группы;
- 3) третий шаг: $d = 2 \div 2 \Rightarrow d = 1$; сортируются все 8 элементов (т. е. формируется одна группа из всех элементов массива).

Сортировка на каждом этапе работы этого алгоритма производится до тех пор, пока не будут упорядочены все элементы внутри группы.

Исходный массив A :

1	2	3	4	5	6	7	8
3	-4	2	0	-1	5	7	-2

Шаг 1:

-1	-4	2	-2	3	5	7	0
----	----	---	----	---	---	---	---

Шаг 2:

-1	-4	2	-2	3	0	7	5
----	----	---	----	---	---	---	---

Шаг 3:

-4	-2	-1	0	2	3	5	7
----	----	----	---	---	---	---	---

Программная реализация:

```
program ShellSort;
const
  N = 8;
```



```
var
  A, B: array[1.. N] of integer;
  c, i, j, d, pp, len, l: integer;
  k: boolean;
begin
  for i := 1 to N do
    write(' ', A[i]);
  writeln;
  d := N div 2;
  while d > 0 do
  begin
    k := true;
    while k do
    begin
      k := false;
      for i := 1 to N - d do
      begin
        if a[i] > a[i + d] then
        begin
          c := a[i];
          a[i] := a[i + d];
          a[i + d] := c;
          k := true;
        end;
      end;
    end;
  end;
  for i := 1 to N do
    write(' ', A[i]);
  writeln;
  d := d div 2;
end;
for i := 1 to N do
  write(' ', A[i]);
writeln;
end.
```

5.9.5. Алгоритмы обработки массивов

Чаще всего при работе с массивами приходится решать простейшие задачи, связанные с поиском минимума (максимума) в массиве, среднего значения элементов массива и т. д. Ниже приведены программы, демонстрирующие решение указанных задач. Программы

MaxElement1d и **MaxElement2d** реализуют поиск максимального числа в одномерном и двумерном массивах, а также подсчет количеств элементов массивов, равных максимальным. Программы **Avr1d** и **Avr2d** предназначены для подсчета среднего значения элементов одномерного и двумерного массивов.

Описание переменных, используемых в программах:

- 1) i, j — переменные, используемые при организации циклов;
- 2) $ind, IndI, IndJ$ — индексы максимальных элементов одномерного и двумерного массива;
- 3) a — массив;
- 4) avr — среднее значение элементов массива.

Алгоритм работы программы **MaxElement1d**:

- 1) заполнение случайными числами массива a ;
- 2) инициализация переменных (предполагается, что максимальный элемент в массиве — первый: $max = a[1]$; $ind = 1$);
- 3) анализ значений элементов массива в цикле — если некоторый элемент массива больше значения переменной max , то в переменную max записывается значение данного элемента и запоминается его индекс:


```

if a[i] > max then
    begin
        max := a[i];
        ind := i;
    end;
      
```
- 4) третий шаг выполняется для всех элементов массива; в результате в переменной max будет находиться значение максимального элемента массива, а в переменной ind — его индекс;
- 5) подсчет количества элементов массива, равных максимальному: если i -й элемент равен максимальному, то переменная amn увеличивается на единицу:


```

for i := 1 to n do
    if a[i] = max then
        amn := amn + 1;
      
```

```

program MaxElement1d;
const
    N = 10;
var
    i, max, ind, amn: byte;
    a: array [1..N] of byte;
  
```

```
begin
  for i := 1 to n do
    a[i] := Trunc(Random(10));
  ind := 1;
  max := a[1];
  for i := 2 to n do
    if a[i] > max then
      begin
        max := a[i];
        ind := i;
      end;
  amn := 0;
  for i := 1 to n do
    if a[i] = max then
      amn := amn + 1;
  writeln('Макс. элемент = ', max);
  writeln('Номер макс. элем. = ', ind);
  writeln('Колич. максим. = ', amn);
  readln;
end.

program MaxElement2d;
const
  n = 4;
  m = 5;
var
  i, j, max, IndI, IndJ, amn: byte;
  a: array [1..n, 1..m] of byte;
begin
  for i := 1 to n do
    for j := 1 to m do
      a[i, j] := Trunc(Random(10));
  max := a[1,1];
  IndI := 1;
  IndJ := 1;
  for i := 1 to n do
    for j := 1 to m do
      if a[i, j] > max then
        begin
          max := a[i, j];
          IndI := i;
```

```

        IndJ := j;
    end;
    amn := 0;
    for i := 1 to n do
        for j := 1 to m do
            if a[i, j] = max then
                amn := amn + 1;
            writeln('Макс. элемент = ', max);
            writeln('Коорд. макс. элем. = (' , IndI, ', ', ' ,
                IndJ, ') ');
            writeln('Колич. максим. = ', amn);
        readln;
    end.

```

Приведенные выше программы поиска максимального элемента массива и подсчета количества элементов, равных максимальному, могут быть легко переделаны для поиска минимума и подсчета количества элементов массива, равных минимальному. Для этого достаточно изменить знак в условном операторе, а также заменить имя переменной *max* на *min* (для смыслового соответствия).

Алгоритм работы программы **Avr1d**:

- 1) заполнение случайными числами массива *a*;
- 2) суммирование элементов массива — данная операция выполняется с помощью цикла, а результат суммирования заносится в переменную *avr*;
- 3) вычисление среднего значения — сумма элементов делится на количество элементов массива ($avr = avr / n$);).

```

program Avr1d;
const
    n = 10;
var
    i, j: byte;
    avr: single;
    a: array [1..n] of single;
begin
    for i:=1 to n do
        a[i] := Random(100) / 10;
    avr := 0;
    for i := 1 to n do
        avr := avr + a[i];

```

```
    avr := avr / n;  
    writeln('Среднее = ', avr:5:3);  
    readln;  
end.  
  
program Avr2d;  
const  
    n = 10;  
    m = 9;  
var  
    i, j: integer;  
    avr: single;  
    a: array [1..n, 1..m] of single;  
begin  
    for i := 1 to n do  
        for j := 1 to m do  
            a[i, j] := Random(100)/10;  
    avr := 0;  
    for i := 1 to n do  
        for j := 1 to m do  
            avr := avr + a[i, j];  
    avr := avr / (n * m);  
    writeln('Среднее = ', avr:5:3);  
    readln;  
end.  
  
program MinElement1d;  
const  
    n = 10;  
var  
    i, min: byte;  
    a: array [1..n] of byte;  
begin  
    for i := 1 to n do  
        a[i] := Random(10);  
    min := a[1];  
    for i := 2 to n do  
        if a[i] < min then  
            min := a[i];  
    writeln('Мин. элемент = ', min);  
    readln;  
end.
```

5.9.6. Генерация последовательностей псевдослучайных чисел

Случайные числа используются в различных алгоритмах, например при шифровании данных, когда от неповторяемости последовательности случайных чисел зависит стойкость шифра.

Источники для получения настоящих случайных чисел найти достаточно трудно. Примером таких источников случайных чисел являются некоторые физические процессы, например тепловой шум. В связи с этим обычно говорят о *генераторе псевдослучайных чисел*. Одним из простейших таких генераторов является линейный конгруэнтный генератор, предложенный Д. Г. Лехмером в 1949 г. Этот алгоритм имеет четыре параметра:

- 1) m — модуль (основание системы, $m > 0$);
- 2) a — множитель ($0 \leq a < m$);
- 3) c — приращение ($0 \leq c < m$);
- 4) X_0 — начальное значение ($0 \leq X_0 < m$).

Множество случайных чисел $\{X_n\}$ вычисляется с помощью формулы:

$$X_{n+1} = (a \cdot X_n + c) \bmod m.$$

Если m , a и c — целые числа, то создается последовательность целых чисел в диапазоне $0 \leq X_n < m$. Выбранные значения a , c и m определяют качество работы генератора. Обычно для генерации большого набора случайных чисел m выбирается приблизительно равным максимальному положительному целому числу, которое можно записать в памяти компьютера.

Если m является простым и $c = 0$, то при определенном значении a период, создаваемый приведенной выше функцией, будет равен $(m - 1)$. Для 32-битной арифметики этому соответствует простое значение $m = 2^{31} - 1$. Таким образом, мы получим для генератора псевдослучайных чисел следующую формулу:

$$X_{n+1} = (a \cdot X_n) \bmod (2^{31} - 1).$$

В результате исследований описываемого алгоритма было установлено, что одно из лучших значений a должно выбираться равным $7^5 = 16807$.

Чтобы генерируемая псевдослучайная последовательность была непредсказуемой, ее можно *модифицировать*. Один из способов такой модификации заключается в использовании в качестве X_0 значения текущего времени в секундах. Другой способ состоит в прибавлении значения текущего времени к каждому числу генерируемой последовательности.

Задача 1. В программе используется одномерный целочисленный массив A с индексами от 0 до 10. Ниже представлен фрагмент программы, записанный на разных языках программирования, в котором значения элементов сначала задаются, а затем меняются:

```

for i := 0 to 10 do
  A[i] := i;
for i := 0 to 10 do
  begin
    A[10-i] := A[i];
    A[i] := A[10-i];
  end;

```

Чему будут равны элементы этого массива после выполнения фрагмента программы [7]?

Решение.

В программе содержится два цикла. В первом цикле i -му элементу массива A присваивается значение индекса этого элемента. Во втором цикле вычисляются значения элементов массива A . Этот процесс отражен в табл. 5.5; индексы элементов, изменяющихся на очередной итерации цикла, выделены в ней жирным шрифтом. В данной таблице приведены только первые 7 итераций цикла, поскольку после выполнения первых шести шагов значения элементов массива A уже изменяться не будут.

Таблица 5.5

Процесс выполнения алгоритма

индексы элементов→	0	1	2	3	4	5	6	7	8	9	10
исходный массив→	0	1	2	3	4	5	6	7	8	9	10
$i = 0$	0	1	2	3	4	5	6	7	8	9	0
$i = 1$	0	1	2	3	4	5	6	7	8	1	0
$i = 2$	0	1	2	3	4	5	6	7	2	1	0
$i = 3$	0	1	2	3	4	5	6	3	2	1	0
$i = 4$	0	1	2	3	4	5	4	3	2	1	0
$i = 5$	0	1	2	3	4	5	4	3	2	1	0
$i = 6$	0	1	2	3	4	5	4	3	2	1	0

Таким образом, после применения к исходному массиву приведенного в условии задачи алгоритма будет сформирован массив, показанный в нижней строке табл. 5.5.

Ответ: массив 0 1 2 3 4 5 4 3 2 1 0.

Задача 2. Значения двух массивов $A[1..100]$ и $B[1..100]$ задаются с помощью следующего фрагмента программы:

```
for n := 1 to 100 do
  A[n] := (n - 80) * (n - 80);
for n := 1 to 100 do
  B[101-n] := A[n];
```

Какой элемент массива B будет наибольшим [5]?

Решение.

Рассмотрим процесс вычисления значений элементов массива A :

```
n = 1 ⇒ A[1] := (1 - 80) · (1 - 80) = 79 · 79;
n = 2 ⇒ A[2] := (2 - 80) · (2 - 80) = 78 · 78;
...
```

Таким образом, с увеличением n значения элементов массива A будут уменьшаться. Следовательно, максимальным в массиве A будет значение его первого элемента.

Значения элементов массива B вычисляются на основе значений элементов массива A следующим образом:

```
n = 1 ⇒ B[101 - 1] := A[1];
n = 2 ⇒ B[101 - 2] := A[2];
...
```

Таким образом, в элемент массива B с индексом 100 записывается значение первого элемента массива A , который, как нам уже известно, является максимальным. Следовательно, наибольшим будет элемент $B[100]$.

Ответ: $B[100]$.

Задача 3. Дан фрагмент программы, обрабатывающей двумерный массив A размера $n \times n$.

```
k := 1;
for i := 1 to n do
  begin
    c := A[i, i];
    A[i, i] := A[k, i];
    A[k, i] := c;
  end;
```

Представим массив в виде квадратной таблицы, в которой для элемента массива $A[i, j]$ величина i является номером строки, а величина j — номером столбца, в котором расположен элемент. Что данный алгоритм меняет местами [6]?

Решение.

Рассмотрим действия, выполняемые на каждой итерации цикла:

- 1) $c = A[i, i]$ — в переменную c записывается i -й элемент главной диагонали массива A ;
- 2) $A[i, i] = A[k, i]$ — в i -й элемент главной диагонали массива A записывается i -й элемент первой строки ($k = 1$) массива A ;
- 3) $A[k, i] = c$ — в i -й элемент первой строки записывается i -й элемент главной диагонали.

Таким образом, указанный алгоритм меняет местами элементы диагонали и k -й строки таблицы.

Ответ: элементы диагонали и k -й строки таблицы.

Задача 4. Дан фрагмент программы, заполняющей двумерный массив A размера $n \times n$:

```
for i := 1 to n do
  for j := 1 to n do
    A[i, j] := i - j + 1;
```

Сколько элементов массива A будет равно 2?

Решение.

Значение элемента массива A равно 2 только в том случае, если выполнится условие: $i - j + 1 = 2$. В этом случае для некоторого i значение j будет вычисляться по формуле: $j = i - 1$. Следовательно, элемент массива A будет равен 2, если он будет находиться на диагонали, находящейся под главной диагональю массива. К таким элементам относятся все элементы массива с координатами $A[i, i-1]$. Причем, как известно, количество элементов диагонали, находящейся под главной (для массива размера $n \times n$), вычисляется по формуле: $n_d = n - 1$, где n — количество элементов главной диагонали.

Ответ: $n - 1$.

Задача 5. Определите, упорядочивает ли фрагмент алгоритма:

```
for k := 1 to 3 do
  if x[k] > x[7-k] then
    begin
      S := x[k];
      x[k] := x[7-k];
      x[7-k] := S;
    end;
```

массив (100, 40, 60, 60, 80, 20).

Решение.

В приведенном алгоритме значение переменной k изменяется от 1 до 3. В условии сравниваются пары элементов данного массива:

- 1) 1-й и 6-й ($7 - 1 = 6$);
- 2) 2-й и 5-й ($7 - 2 = 5$);
- 3) 3-й и 4-й ($7 - 3 = 4$).

Затем если первый элемент больше шестого, то они меняются местами. Аналогичные действия выполняются и для других пар элементов массива. Учитывая это, рассмотрим исходный массив:

1) в этом массиве первый элемент больше шестого, следовательно, они меняются местами, после чего первый элемент станет минимальным, а шестой — максимальным:

(20, 40, 60, 50, 80, 100)

2) второй элемент меньше пятого, значит, они останутся на своих местах (порядок следования остальных элементов на данном шаге также не будет нарушен):

(20, 40, 60, 50, 80, 100)

3) третий элемент больше четвертого, следовательно, они меняются местами:

(20, 40, 50, 60, 80, 100).

Таким образом, мы получаем упорядоченный массив.

Ответ: данный фрагмент упорядочивает массив.

Задача 6. Все элементы двумерного массива A размером 10×10 элементов первоначально были равны 0. Затем значения этих элементов меняются с помощью вложенного оператора цикла в представленном фрагменте программы:

```
for n := 1 to 4 do
  for k := n to 4 do
    begin
      A[n, k] := A[n, k] + 1;
      A[k, n] := A[k, n] + 1;
    end;
```

Сколько элементов массива в результате будут равны 1 [2]?

Решение.

Отметим, что диапазон значений переменной k зависит от значения переменной n . Выполним фрагмент программы по шагам ($n = 1$, $k = 1 \dots 4$):

- 1) $k = 1 \Rightarrow A[1,1] = A[1,1] + 1$, то $A[1,1] = 1$; $A[1,1] = A[1,1] + 1$, то $A[1,1] = 2$;

- 2) $k = k + 1$, т. е. $k = 2 \Rightarrow A[1,2] = A[1,2] + 1$, то $A[1,2] = 1$;
 $A[2,1] = A[2,1] + 1$, то $A[2,1] = 1$;
 3) $k = k + 1$, т. е. $k = 3 \Rightarrow A[1,3] = A[1,3] + 1$, то $A[1,3] = 1$;
 $A[3,1] := A[3,1] + 1$, то $A[3,1] = 1$;
 4) $k = k + 1$, т. е. $k = 4 \Rightarrow A[1,4] = A[1,4] + 1$, то $A[1,4] = 1$;
 $A[4,1] = A[4,1] + 1$, то $A[4,1] = 1$;

На каждой итерации цикла по n при этом выполняются следующие действия:

- 1) вычисляется значение диагонального элемента (получаем, что значение диагонального элемента равно 2);
- 2) заполняются единицами строка и столбец, в начале которых стоит указанный диагональный элемент (заполняются единицами все элементы первой строки, начиная со второго элемента и заканчивая четвертым, а также все элементы первого столбца со второго по четвертый).

Результат первой
итерации цикла n

	1	2	3	4	5
1	2	1	1	1	0
2	1	0	0	0	0
3	1	0	0	0	0
4	1	0	0	0	0
5	0	0	0	0	0

Результат работы
алгоритма

	1	2	3	4	5
1	2	1	1	1	0
2	1	2	1	1	0
3	1	1	2	1	0
4	1	1	1	2	0
5	0	0	0	0	0

Индексы циклов по n и по k изменяются от 1 до 4, поэтому в результате выполнения алгоритма будет обработана только часть матрицы размером 4×4 . Анализируя полученные результаты, можно заметить, что значения диагональных элементов этой части матрицы будут установлены в 2, а значения всех оставшихся элементов указанной части матрицы станут равными 1. Остальные элементы массива будут равны 0. Таким образом, в результате работы алгоритма количество единиц в массиве станет равно 12.

Ответ: 12.

Задача 7. Дан целочисленный массив из 30 элементов. Элементы этого массива могут принимать целые значения от 0 до 100 (баллы учащихся выпускного класса за итоговый тест по информатике). Для получения положительной оценки за тест требовалось набрать не менее 20 баллов. Опишите на русском языке или на одном из известных вам языков программирования алгоритм, который находит и выводит минимальный балл среди учащихся, получивших за тест

положительную оценку. (Известно, что в классе хотя бы один учащийся получил за тест положительную оценку.)

Исходные данные объявлены так, как показано ниже. Запрещается использовать переменные, не описанные ниже, но разрешается не использовать часть из них.

```

const
    N = 30;
var
    a: array [1..N] of integer;
    i, j, min: integer;
begin
    for i := 1 to N do
        readln(a[i]);
    ...
end.

```

В качестве ответа необходимо привести фрагмент программы, который должен находиться на месте многоточия [7].

Решение.

Для решения этой задачи необходимо записать алгоритм поиска учащегося, который получил положительную оценку и балл которого минимален, среди всех учащихся, получивших положительную оценку. Предположим, что введен следующий массив исходных данных:

№ учащегося	1	2	3	4	5	6	7
Балл	25	30	10	12	11	22	43

Тогда из всех учащихся, перечисленных в этой таблице, положительную оценку получают учащиеся №№ 1, 2, 6 и 7 (у которых количество баллов не менее 20), причем минимальное количество баллов набрал учащийся под номером 6.

Данная задача аналогична задаче поиска минимального элемента массива, однако необходимо учитывать, что в нашем случае при поиске минимального элемента необходимо использовать сложное условие, в котором учитывается, что искомый учащийся не может набрать количество баллов меньше 20:

```

program FindMin;
const
    N = 30;
var

```

```
    a: array [1..N] of integer;
    i, j, min: integer;
begin
    for i := 1 to N do
        readln(a[i]);
    min := a[1];
    for i := 2 to N do
        if (a[i] < min) and (a[i] >= 20) then
            min := a[i];
    writeln('min = ', min);
    readln;
end.
```

Задача 8. Опишите на русском языке или одном из языков программирования алгоритм подсчета количества уникальных элементов в целочисленном массиве длины $N < 100$. Элементы данного массива могут принимать значения от 0 до 49.

Решение.

Этапы решения данной задачи:

- 1) заполнение исходного массива случайными числами;
- 2) подсчет, сколько раз встречается каждое значение элемента в исходном массиве;
- 3) определение количества уникальных элементов.

Для решения задачи введем массив *hist*, состоящий из 50 элементов:

```
hist: array [0..NH-1] of byte;
```

и массив *a*, в котором хранятся введенные данные:

```
a: array [1..N] of byte;
```

Значение *i*-го элемента массива *hist* показывает, какое количество элементов исходного массива равно *i*. Если же *i*-й элемент массива *hist* равен единице, то значение исходного элемента является уникальным.

Подсчет количества уникальных элементов, таким образом, выполняется в два этапа:

- 1) определяется, сколько раз встречается каждое число в массиве *a*:

```
for i := 1 to N do
    hist[a[i]] := hist[a[i]] + 1;
```

- 2) определяется количество уникальных элементов:

```
for i := 0 to NH-1 do
    if hist[i] = 1 then
        NumUniq := NumUniq + 1;
```

Поставленную задачу решает программа **CalcUnique**:

```

program CalcUnique;
const
    N = 99;
    NH = 50;
var
    a: array [1..N] of byte;
    hist: array [0..NH-1] of byte;
    i: word;
    NumUniq: byte;
begin
    NumUniq := 0;
    for i := 1 to N do
        a[i] := random(100);
    for i := 0 to NH-1 do
        hist[i] := 0;
    for i := 1 to N do
        hist[a[i]] := hist[a[i]] + 1;
    for i := 0 to NH-1 do
        if hist[i] = 1 then
            NumUniq := NumUniq + 1;
    writeln('Количество уникальных элементов = ',
           NumUniq);
    readln;
end.

```

Задача 9. Опишите на русском языке или на одном из языков программирования алгоритм получения из заданного целочисленного массива размером 30 элементов другого массива, который будет содержать модули значений элементов первого массива (не используя специальной функции, вычисляющей модуль числа) [6].

Решение.

Введем обозначения: a — исходный массив; b — результирующий массив.

Введем правила вычисления значений элементов массива b по элементам массива a :

1) если i -й элемент массива a — отрицательный ($a[i] < 0$), то значение i -го элемента массива b вычисляется следующим образом: $b[i] = -1 \cdot a[i]$;

2) если i -й элемент массива a — положительный ($a[i] > 0$), то значение i -го элемента массива b просто копируется из массива a : $b[i] = a[i]$.

Описанный алгоритм реализован в программе **BuildAbs**:

```
Program BuildAbs;
const
  N = 30;
var
  a, b: array [1..N] of integer;
  i: integer;
begin
  for i := 1 to N do
    a[i] := random(100) - 20;
  for i := 1 to N do
    if a[i] < 0 then
      b[i] := -a[i]
    else
      b[i] := a[i];
  end.
```

Задача 10. Опишите на русском языке или на одном из языков программирования алгоритм подсчета максимального количества подряд идущих совпадающих элементов в целочисленном массиве длины 30 [5].

Решение.

Введем обозначения:

- 1) *MaxCep* — переменная, используемая для хранения максимального количества подряд идущих совпадающих элементов;
- 2) *NumCep* — переменная, используемая для хранения количества элементов в последней группе совпадающих элементов;
- 3) *a* — исходный массив.

Рассмотрим алгоритм решения задачи. Сравниваем *i*-й и (*i*+1)-й элементы массива *a*:

- 1) если $a[i] = a[i+1]$, то увеличиваем *NumCep* на единицу;
- 2) если $a[i] \neq a[i+1]$, то сравниваем текущее значение *NumCep* со значением переменной *MaxCep*, и если $MaxCep > NumCep$, то записываем в переменную *MaxCep* значение переменной *NumCep*, после чего записываем в счетчик *NumCep* единицу.

Далее указанным образом в цикле выполняется анализ всего массива.

После завершения цикла нужно еще раз сравнить значение счетчика *NumCep* со значением переменной *MaxCep* и переопределить ее, если счетчик окажется больше (см. программу **MaxLen**).

```
Program MaxLen;
const
  N = 30;
var
  a: array [1..N] of integer;
  MaxCep, NumCep, i: integer;
begin
  MaxCep := 1;
  NumCep := 1;
  for i := 1 to N do
    a[i] := random(100);
  for i:= 2 to N do
    begin
      if a[i] = a[i-1] then
        NumCep := NumCep + 1
      else
        begin
          if NumCep > MaxCep then
            MaxCep := NumCep;
          NumCep := 1;
        end;
      end;
    if NumCep > MaxCep then
      MaxCep := NumCep;
    writeln(MaxCep);
    readln;
  end.
```

Задача 11. Опишите на русском языке или на одном из языков программирования алгоритм поиска номера первого из двух последовательных элементов в целочисленном массиве из 30 элементов, сумма которых максимальна (если таких пар несколько, то можно выбрать любую из них) [4].

Решение.

Введем обозначения:

- 1) *max* — сумма двух последовательных элементов;
- 2) *e* — номер первого элемента из двух последовательных.

Поиск указанных пар выполняется в цикле перебором всех элементов массива. Значение переменной-счетчика при этом изменяется в диапазоне от 1 до 29, так как для последнего элемента нет «пары». Алгоритм же поиска номера первого из двух последовательных элементов подобен алгоритму поиска номера максимального элемента массива (см. программу **MaxElement1d**).


```
program FindPairs;
const
  N = 30;
var
  a: array [1..N] of integer;
  max, i, ind: integer;
begin
  for i := 1 to N do
    a[i] := random(100);
  max := a[1] + a[2];
  ind := 1;
  for i := 1 to N - 1 do
    if (max < (a[i] + a[i+1])) then
      begin
        max := a[i] + a[i+1];
        ind := i;
      end;
  writeln('ind = ', ind);
  readln;
end.
```

5.10. Символьные типы данных

5.10.1. Тип char

Величина (переменная или константа) типа `char` предназначена для хранения одного символа из таблицы ASCII. Каждому символу при этом соответствует неотрицательное число, называемое *кодом символа*.

Объявление переменной типа `char`:

```
var
  c: char;
```

Инициализацию величины типа `char` можно выполнить несколькими способами:

1) **const**

```
c = 'f';
```

2) **var**

```
c: char;
```

```
begin
```

```
c := 'f';
```

```
end.
```

```
3) var
    c: char;
begin
    c := #102;
    //код символа 'f'
end.
4) var
    c: char;
begin
    write('c = ');
    readln(c);
end.
```

Отметим, что в третьем способе значение переменной типа `char` задается указанием числового кода символа: `c := #номер_кода`.

Для работы с символами и их кодами используются следующие функции:

1) `ord(c: char): byte` – возвращает код символа, хранящегося в переменной `c`;

2) `chr(code: byte): char` – возвращает символ, код которого указан в переменной `code`.

Пример:

```
program CharCode;
var
    c, c1, c2: char;
    code: byte;
begin
    c1 := 'A';
    c2 := #69;
    write('Введите символ = ');
    readln(c);
    code := ord(c);
    writeln('Введен символ = ', c, '. ASCII-код = ',
            code);
    writeln('Введите ASCII-код:');
    readln(code);
    c := chr(code);
    writeln('Коду ', code, ' соответствует символ ', c);
    readln;
end.
```

Величины типа `char` можно использовать в операциях сравнения:

```
program CharRelation;
var
  res1, res2: boolean;
  c1, c2, c3, c4: char;
begin
  c1 := 'b';
  c2 := 'x';
  res1 := c1 < c2;
  c3 := #12;
  c4 := #150;
  res2 := c3 > c4;
  writeln('"'b'" > "'c'" = ', res1);
  writeln('#12 > #150 = ', res2);
  readln;
end.
```

В результате выполнения программы **CharRelation** в переменную *res1* записывается значение `true`, так как код символа 'b' меньше кода символа 'x', а в переменную *res2* — значение `false`, так как код символа, хранящийся в переменной *c3*, меньше кода, хранящегося в *c4*. Таким образом, результат выполнения этой программы:

```
'b' > 'c' = TRUE
#12 > #150 = FALSE
```

Пример проверки принадлежности символа определенному диапазону показан в программе **CharCheck**:

```
program CharCheck;
var
  c: char;
begin
  write('Введите символ = ');
  readln(c);
  if (c >= '0') and (c <= '9') then
    writeln('Введена цифра')
  else
    writeln('Введена не цифра');
  readln;
end.
```

Преобразование строчного символа в прописной выполняет функция `UpCase(c: char): char` (см. программу **CharUp**). Если *c* — строчная латинская буква, то данная функция возвращает со-

ответствующую прописную латинскую букву; в противном случае символ, хранящийся в *c*, возвращается без изменения (с русскими буквами данная функция не работает).

```
program CharUp;
var
  c: char;
begin
  write('Введите символ = ');
  readln(c);
  c := UpCase(c);
  writeln(c);
  readln;
end.
```

5.10.2. Тип `string`

Величина типа `string` предназначена для хранения последовательности символов. Количество символов в такой последовательности называется *длиной строки*.

Определение переменной типа `string` выглядит следующим образом.

```
var
  s: string[n];
```

Длина строки (*n*) — целое число в диапазоне 1 .. 255. Если данный параметр не указывается, то по умолчанию он принимается равным 255, например:

```
var
  s: string;
```

Строковые константы записываются как последовательности символов, ограниченные апострофами (одинарными кавычками). Допускается также формирование строк с использованием записи символов их десятичными кодами (в виде комбинации # и кода символа) и управляющих символов (комбинации ^ и некоторых заглавных латинских букв).

Ниже приведены примеры инициализации величины (переменной или константы) типа `string`:

```
1) const
  str1 = 'строка';
  str2 = #54#32#61;
  str3 = 'abcde'^A^M;
```

```
2) var
   s: string;
   begin
     s := 'string';
   end.
3) var
   s: string;
   begin
     s := #102#102;
   end.
4) var
   s: string;
   begin
     readln(s);
   end.
```

Существует одна особенность при вводе строки с клавиатуры: признаком окончания ввода строки является нажатие клавиши **Enter**, поэтому все следующие за ней переменные необходимо вводить с новой строки. Поэтому для ввода строки лучше всего использовать отдельную процедуру `readln`.

Доступ к i -му символу строки s можно получить следующим образом. Допустим, что в строке s ($s: \text{string}[3]$) хранится значение 'str', тогда в $s[1]$ находится символ 's', в $s[2]$ — 't', а в $s[3]$ — 'r'. Таким образом, в $s[i]$ хранится i -й символ строки, где нумерация элементов массива-строки начинается с единицы.

Пустая строка обозначается двумя подряд стоящими апострофами. А чтобы использовать символ апострофа в самой строке, его необходимо удвоить:

```
const
  x = ''';
```

Строковые переменные с разными длинами можно присваивать друг другу. При этом при записи в строковую переменную значения, размер которого превышает размер переменной, лишние символы будут отброшены. Величины же типа `char` можно присваивать любым строковым переменным (т. е. `char` рассматривается как строка из одного символа).

Над строками определены следующие операции:

1) операция слияния (*конкатенации*), обозначаемая знаком «+» (см. программу **StrConCat**):

```
program StrConCat;
var
  a, b, c: string;
begin
  a := 'Pas';
  b := 'cal';
  c := a + b;
  writeln('a = ', a);
  writeln('b = ', b);
  writeln('c = ', c);
  readln;
end.
```

2) операции сравнения: <, >, =, <>, <=, >=. Две строки сравниваются посимвольно, слева направо, по кодам их символов. Если одна строка меньше другой по длине, то недостающие символы условно заменяются символами с кодом 0. Например, в результате работы программы **StrCompare** в переменную *res* будет записано `false`, так как 1-й, 2-й, 4-й, 5-й, 6-й и 7-й символы строк *s1* и *s2* одинаковы, но код третьего символа строки *s1* меньше кода третьего символа строки *s2*:

```
program StrCompare;
var
  s1, s2, s3: string;
  res: boolean;
begin
  s1 := 'syMbol';
  s2 := 'symbol';
  res := s1 > s2;
  writeln('syMbol > symbol = ', res);
  readln;
end.
```

Функции и процедуры, используемые для обработки строковых величин:

1) `length(s: string): byte` — вычисляет длину строки *s*, например:

```
n := length('Pascal');
```

В результате выполнения этой функции в *n* будет записано число 6;

2) `pos(substr, s: string): byte` — выполняет поиск в строке *s* подстроки *substr*. Данная функция возвращает номер символа, с которого подстрока начинается в строке. Если же подстрока не найдена, то функция возвращает значение 0. Пример:

```
s := 'Free Pascal';  
p1 := pos('Pascal', s);  
p2 := pos('Turbo', s);
```

В результате выполнения указанных операторов в *p1* будет записано число 6, а в *p2* — 0;

3) `copy(s: string; index: integer; count: integer): string` — возвращает подстроку длиной *count* символов, выделенную из исходной строки *s*, начиная с символа под номером *index*.

Пример:

```
s := 'Free Pascal';  
s2 := copy(s, 1, 4);  
s3 := copy(s, 6, 6);
```

В результате выполнения указанных операторов в *s1* будет записано значение 'Free', а в *s2* — 'Pascal';

4) `delete(var s: string; index, count: integer)` — удаляет из строки *s* подстроку длиной *count* символов, начиная с символа под номером *index*. Пример:

```
s := 'Free Pascal';  
delete(s, 1, 5);
```

После выполнения указанных операторов в *s* будет записано значение 'Pascal';

5) `insert(source: string; var s: string; index: integer)` предназначена для вставки строки *source* в строку *s*, начиная с символа *index* этой строки. Пример:

```
s := 'Pascal';  
insert('Free ', s, 1);
```

В результате в *s* будет записано значение 'Free Pascal';

6) `val(s: string; x: <числ. тип.>; var code: integer)` — преобразует строковую запись числа, содержащуюся в *s*, в числовое представление, помещая результат в *x* (где *x* — целая или вещественная переменная). Если при этом в *s* встречается недопустимый (по правилам записи чисел) символ, то преобразование не происходит, а в *code* записывается позиция первого недопустимого символа; выполнение программы при этом не прерывается. Если же после выполнения процедуры значение *code* равно 0, то это свидетельствует об успешном выполнении преобразования;

7) `str(x: <числ. перем.>, var s: string)` — преобразует число, хранящееся в *x*, в его символьное представление, которое записывается в *st*;

8) `concat(s1, [s2, ..., sn]: string): string` — выполняет объединение строк, указанных в качестве параметров. Данная функция эквивалентна операции конкатенации (+). Строки в результате записываются в том порядке, в котором они перечислены в параметрах функции. Каждый параметр должен представлять собой выражение строкового типа. Ограничение на количество соединяемых строк отсутствует, но если длина результата превышает 255 символов, то он усекается до 255 символов. Пример:

```
s1 := 'ab ';  
s2 := 'cd';  
res := concat(s1, s2);
```

В результате выполнения указанных операторов в *res* будет записана строка 'ab cd'.

Задача 1. Разработать программу замены во введенной строке латинской буквы 'S' на пробел.

Решение.

Решение этой задачи основано на поэлементной обработке строки: каждый ее символ сравнивается с символом 'S': если *i*-й символ строки равен 'S', то данный символ заменяется на пробел:

```
program LetterReplace;  
var  
  s: string;  
  i: integer;  
begin  
  writeln('Введите текст:');  
  readln(s);  
  for i:=1 to length(s) do  
    if (s[i] = 'S') then  
      s[i] := ' ';  
  writeln(s);  
  readln;  
end.
```

Задача 2. Разработать программу подсчета количества заданных букв в тексте.

Решение.

Алгоритм решения задачи:

- 1) искомые буквы задаются в виде строки-шаблона;
- 2) выбирается очередная буква исходной строки;

3) проверяется, входит ли выбранный символ в строку-шаблон; если это условие выполнено, то количество найденных букв увеличивается на 1.

Данный алгоритм выполняется до тех пор, пока не будут проанализированы все буквы введенной строки.

```
program LetterStatistic;
var
  s, pattern: string;
  i, counter: integer;
begin
  pattern := 'абс';
  writeln('Введите текст:');
  readln(s);
  counter := 0;
  for i := 1 to length(s) do
    if (pos(s[i], pattern) <> 0) then
      counter := counter + 1;
  writeln('Количество искомых букв в тексте = ',
    counter);
  readln;
end.
```

Задача 3. Разработать программу подсчета во введенной строке количества слов длиной 5 символов.

Решение.

Известно, что слова в тексте разделяются пробелами. Пусть между словами стоит только один пробел, тогда задачу можно решить, используя следующий алгоритм:

1) определить с помощью функции `pos` номер (индекс) первого пробела в строке;

2) скопировать во временную строку (`wrd`) часть исходной строки с первого символа до пробела (т. е. i -е слово строки);

3) вычислить длину полученного слова с помощью функции `length`;

4) если длина слова равна 5 символам, то увеличить значение счетчика слов на 1;

5) удалить из исходной строки i -е слово и пробел; перейти на шаг 1.

Анализ исходной строки выполняется согласно вышеприведенному алгоритму до тех пор, пока в строке не останется символов.

Примечание. До начала анализа к введенной строке дописывается один пробел во избежание заикливания.

```
program WordCount;
var
  s, wrd: string;
  p, counter: integer;
begin
  writeln('Введите строку:');
  counter := 0;
  readln(s);
  s := s + ' ';
  while s <> '' do
    begin
      p := pos(' ', s);
      wrd := copy(s, 1, p-1);
      writeln(wrd);
      delete(s, 1, p);
      if (length(wrd) = 5) then
        counter := counter + 1;
    end;
  writeln('Количество слов длиной 5 символов = ',
    counter);
  readln;
end.
```

5.11. Записи

Запись — это сложная структура данных, состоящая из фиксированного числа элементов, называемых *полями записи*, которые могут относиться к различным типам.

Определение переменной типа запись:

```
var
  <имя_записи>: record
    <список полей>;
  end;
```

Описание списка полей:

```
<имя поля 1>: <тип поля 1>;
...
<имя поля N>: <тип поля N>;
```

Список полей состоит из имен полей, после каждого из которых следует двоеточие и указывается тип соответствующего поля. Ниже приведен пример описания записи координат точки на плоскости:

```
var
  coord: record
    x: single;
    y: single;
  end;
```

Доступ к элементам записи выполняется следующим образом:

<имя записи>.<имя поля>.

После определения переменной `coord` можно присвоить полям записи требуемые значения:

```
begin
  coord.x := 2;
  coord.y := 3;
end.
```

Можно также определить тип — запись:

```
type
  <имя_записи> = record
    <список полей>;
  end;
```

после чего вводить переменные данного типа:

```
program RecType;
type
  tcoord = record
    x: single;
    y: single;
  end;
var
  p1, p2: tcoord;
begin
  p1.x := 2;
  p1.y := -2;
  p2.x := 10;
  p2.y := -4;
end.
```

Непосредственно к записям применять какие-либо операции нельзя. Их можно применять только к отдельным полям записи, в зависимости от их типов. Пример:

```
program EnumExample;  
type  
  month = (Jan, Feb, Mar, Apr, May, Jun, Jly, Aug,  
           Sep, Oct, Nov, Dec);  
  date = record  
    d: 1..31;  
    m: month;  
    y: 1900..2030;  
  end;  
var  
  dt1, dt2: date;  
begin  
  dt1.d := 10;  
  dt1.m := jan;  
  dt1.y := 2010;  
  dt2.d := dt1.d + 2;  
  dt2.m := jly;  
  dt2.y := dt2.y + 2;  
end.
```

5.11.1. Массив записей

Записи удобно использовать для хранения и обработки информации о каких-либо объектах. Если требуется хранить информацию о нескольких объектах одного типа, то можно использовать массив записей.

Способы объявления массива записей:

1) массив записей объявляется как переменная:

```
const  
  n = 10;  
var  
  coord: array [1..n] of record  
    x: single;  
    y: single;  
  end;
```

2) сначала объявляется тип — запись, после чего вводится массив элементов указанного типа:

```
const
  n = 10;
type
  tcoord = record
    x: single;
    y: single;
  end;
var
  coord: array [1..n] of tcoord;
```

Доступ к полю элемента массива записей осуществляется следующим образом:

<имя_массива>[<индекс_элемента>].<имя_поля>

Пример:

```
coord[1].x := -10;
```

5.11.2. Инициализация записи

При заполнении полей записи можно использовать ввод с клавиатуры (процедура `readln`, см. программу **InitRecRead**) или заполнение полей случайными числами (функция `random`, см. программу **InitRecRand**).

```
program InitRecRand;
var
  coord: record
    x: single;
    y: single;
  end;
begin
  coord.x := random;
  coord.y := random;
end.

program InitRecRead;
var
  coord: record
    x: single;
    y: single;
  end;
  x, y: single;
begin
  write('Введите x = ');
```

```

    readln(x);
    write('Введите y = ');
    readln(y);
    coord.x := x;
    coord.y := y;
end.

```

Запись можно инициализировать в разделе **const**:

```

const
    <имя записи>: <тип записи> =
        (<имя поля1>: <знач. поля1>; ... ; <имя поляN>:
        <знач. поляN>);

```

Имя поля от его начального значения при этом отделяется двоеточием; значения соседних полей разделяются точкой с запятой:

```

type
    point = record
        x, y : single;
    end;
const
    p1: point = (x: 0; y: 12);

```

Поля в списке должны указываться в той последовательности, в какой они перечислены в определении типа.

5.11.3. Инициализация массива записей

Заполнение массива записей можно выполнить, используя ввод значений с клавиатуры (см. программу **ReadRecArr**) или генератор случайных чисел (см. программу **RandRecArr**).

```

program RandRecArr;
const
    n = 10;
var
    coord: array [1..n] of record
        x: single;
        y: single;
    end;
    i: byte;
begin
    for i := 1 to n do
        begin

```

```
        coord[i].x := random;
        coord[i].y := random;
    end;
end.

program ReadRecArr;
const
    n = 10;
var
    coord: array [1..n] of record
        x: single;
        y: single;
    end;
    i: byte;
    x, y: single;
begin
    for i := 1 to n do
        begin
            write('Введите x = ');
            readln(x);
            write('Введите y = ');
            readln(y);
            coord[i].x := x;
            coord[i].y := y;
        end;
    end.
end.
```

Существует также еще один способ инициализации массива записей — использование *константы-записи*:

```
const
    <имя записи>: array [<ДИ1>, ..., <ДИN>] of
        <тип записи> = <имя поля1>: <знач. поля1>; ... ;
        <имя поляN>: <знач. поляN>;
```

Пример:

```
type
    data = record
        day: 1..31;
        month: 1..12;
        year: 1000..2100;
    end;
```

```
const
  SomeDays: array[1..3] of data
    = ((day: 8; month: 3; year: 1175),
       ( day: 23; month: 2; year: 1770),
       ( day: 1; month: 9; year: 2010));
```

5.11.4. Оператор with

Оператор **with** используется для упрощения доступа к полям записи. Синтаксис оператора:

```
with <имя_записи> do
  <оператор>;
```

В области действия оператора **with** можно обращаться к полям записи <имя_записи> без указания имени этой записи. Например, ниже в программе **UseWith1** используется обращение к полям записи при помощи оператора **with**, а в программе **UseWith2** — без использования данного оператора.

```
program UseWith1;
var
  coord: record
    x: single;
    y: single;
  end;
begin
  with coord do
    begin
      x := 2;
      y := 3;
    end;
  end.

program UseWith2;
var
  coord: record
    x: single;
    y: single;
  end;
begin
  coord.x := 2;
  coord.y := 3;
end.
```


Если в пределах области действия оператора **with** нужно обратиться к нескольким полям или выполнить несколько операторов, то следует использовать операторные скобки **begin .. end**.

Оператор **with** можно также применять при работе с массивами записей, например:

```
program ArrayWith;
const
  n = 10;
var
  CoordArr: array [1..n] of record
    x: single;
    y: single;
  end;
  i: byte;
begin
  for i := 1 to n do
    with CoordArr[i] do
      begin
        x := random;
        y := random;
      end;
    end;
  end.
```

Предположим теперь, что в разделе **var** объявлена переменная *y*, а также запись, включающая поле с именем *y*. Чтобы в области действия оператора **with** обратиться не к полю записи, а к глобальной переменной с таким же именем, перед этой переменной нужно *указать через точку имя программы*: <имя_программы>.<имя_переменной>. Пример:

```
program TestWith;
var
  y: byte;
  coord : record
    x: single;
    y: single;
  end;
begin
  with coord do
    begin
      x := 2;
```

```

        y := 3;
        TestWith.y := 10;
    end;
    writeln('y = ', y);
end.

```

Существует также возможность определения записи, в которой может изменяться набор полей. Такая запись называется *записью с вариантной частью*:

```

var
  <имя_записи>: record
    case <переключатель>: <тип> of
      <вариант 1>: (<поле>: <тип>; ...);
    ...
      <вариант N>: (<поле>: <тип>; ...);
    end;

```

Вариантная часть начинается с зарезервированного слова **case**, после которого указывается поле записи, которое в дальнейшем будет служить «переключателем». Список значений, входящих в вариант, может быть константой, диапазоном или объединением нескольких констант или диапазонов. Поля, которые должны быть включены в структуру записи при выполнении соответствующего варианта, заключаются в круглые скобки.

Одна из особенностей вариантной части заключается в том, что для каждого варианта полей *отводится одна и та же область памяти*. Например, в программе **VariantRec** описывается запись *tps*, включающая три варианта, где каждый из этих вариантов занимает в памяти один и тот же участок размером в 4 байта. При этом в зависимости от того, к какому полю записи выполняется обращение, этот участок памяти рассматривается как массив из двух элементов типа *word* (поле *wr*), как массив из четырех целых чисел типа *byte* (поле *bt*) или как четыре символа (поле *ch*).

```

program VariantRec;
var
  tps: record
    case num: byte of
      1: (bt: array [0..3] of byte);
      2: (wr: array [0..1] of word);
      3: (ch: array [0..3] of char);
    end;

```

```

begin
  with tps do
    begin
      bt[0] := 70;
      bt[1] := 80;
      bt[2] := 90;
      bt[3] := 100;
      writeln('Word 1 = ', wr[0], '; Word 2 = ', wr[1]);
      writeln('Char 1 = ', ch[0], '; Char 2 = ', ch[1]);
      writeln('Char 3 = ', ch[2], '; Char 4 = ', ch[3]);
    end;
  readln;
end.

```

Например, записи *tps* можно сначала присвоить в качестве значения четыре символа, а затем просмотреть ее по байтам или элементам типа *word*:

byte			
70	80	90	100
word			
20550		25690	
char			
F	P	Z	d

Задача 1. В магазине продают телевизоры и ноутбуки. Требуется описать ассортимент этого магазина с помощью записи, включающей вариантную часть.

Решение.

Общие характеристики товаров: цена, марка и год выпуска.

Уникальные характеристики:

1) для ноутбука — процессор (*CPU*), размер оперативной памяти (*OZU*), операционная система (*OS*);

2) для телевизора — размер диагонали (*DiagSize*) и тип: с электронно-лучевой трубкой (*ELT*), плазменный (*Plazma*) или жидкокристаллический (*LCD*).

Реализация описанной структуры выполняется с использованием записи с вариантной частью:

```

type
  TvType = (ELT, LCD, Plazma);

```

```

product = record
  name: string[20];
  cost: single;
  year: 1980..2010;
  case item: byte of
    1: (CPU: string[10]; OZU: integer; OS: string[30]);
    2: (DiagSize: byte, Ttype: TvType);
  end;

```

Задача 2. Разработать программу, позволяющую хранить и обрабатывать данные об абонентах телефонной сети. Характеристики абонента: фамилия, номер телефона, год рождения.

Решение.

Алгоритм работы программы:

- 1) заполнение массива записей;
- 2) обработка команд пользователя — выполняется анализ нажатой клавиши: 0 — выход из программы, 1 — поиск абонента по телефонному номеру, 2 — вывод отсортированной по телефонному номеру информации об абонентах. Команды обрабатываются с помощью оператора выбора и цикла **repeat**;
- 3) сортировка массива записей по телефонному номеру выполняется на основе метода «пузырька».

```

program RecordsSort;
type
  PhoneBookRec = record
    surname: string[20]; // фамилия
    birthday: word;      // год рождения
    phoneNumber: word;   // номер телефона
  end;
const
  n = 5;
var
  PhoneBook: array [1..n] of PhoneBookRec;
  i, j: byte;
  i_str: string;
  a, b, PhoneNumber: word;
  ch: char;
  founded: boolean;
  t: PhoneBookRec;

begin

```

```
randomize;
a := 1900;
b := 1985;
for i := 1 to n do
  with PhoneBook[i] do
  begin
    str(i, i_str);
    surname := 'Surname' + i_str;
    birthday := round(a + (b - a) * random);
    phoneNumber := round(10000 +
      (60000 - 10000) * random);

  end;
writeln('Записи телефонной книги:');
writeln('Фамилия: Год рожд.: № телефона:');
for i := 1 to n do
  with PhoneBook[i] do
    writeln(surname, birthday:11, phoneNumber:12);
repeat
  writeln('Введите команду:');
  writeln('0 - Выход');
  writeln('1 - Поиск по номеру телефона');
  writeln('2 - Вывод отсортированной книги');
  readln(ch);
case ch of
  '1':
    begin
      write('Введите номер телефона: ');
      readln(phoneNumber);
      founded := false;
      for i := 1 to n do
        if phoneNumber = PhoneBook[i].phoneNumber
        then
        begin
          writeln('Номер телефона найден');
          writeln('Фамилия: Год рожд.: ',
            '№ телефона:');
          with PhoneBook[i] do
            writeln(surname, birthday:11,
              phoneNumber:12);
          founded := true;
          break;
```

```

        end;
        if founded = false then
            writeln('Указанный номер телефона ',
                'не найден');
        end;
    '2':
    begin
        for i := 1 to n - 1 do
            for j := 1 to n - i do
                if PhoneBook[j].phoneNumber <
                    PhoneBook[j + 1].phoneNumber then
                    begin
                        t := PhoneBook[j];
                        PhoneBook[j] := PhoneBook[j + 1];
                        PhoneBook[j + 1] := t;
                    end;
            writeln('Записи отсортированной ',
                'телефонной книги:');
            writeln('Фамилия: Год рожд.: № телефона:');
            for i := 1 to n do
                with PhoneBook[i] do
                    writeln(surname, birthday:11,
                        phoneNumber:12);
            end;
        end;
    until (ch = '0');
end.

```

Задача 3. На вход программе подаются сведения о сдаче экзаменов учениками 9-х классов некоторой средней школы. В первой строке сообщается количество учеников N , которое не меньше 10, но не превосходит 100, а каждая из следующих N строк имеет следующий формат:

<Фамилия> <Имя> <оценки> ,

где <Фамилия> — строка, состоящая не более чем из 20 символов, <Имя> — строка, состоящая не более чем из 15 символов, а <оценки> — записанные через пробел три целых числа, соответствующие оценкам по пятибалльной системе. Записи <Фамилия> и <Имя>, а также <Имя> и <оценки> разделены одним пробелом. Пример входной строки:

Иванов Петр 4 5 3

Требуется написать программу, которая будет выводить на экран фамилии и имена трех худших по среднему баллу учеников. (Если среди остальных есть ученики, набравшие тот же средний балл, что и один из трех худших, то следует также вывести и их фамилии и имена.) [5].

Решение.

Результат работы данной программы — список фамилий и имен учеников, набравших наименьшее количество баллов, поэтому при чтении исходных данных фамилии и имена необходимо сохранять. Также необходимо сохранять суммарное количество баллов, набранных каждым учеником. Для хранения указанной информации о каждом ученике можно использовать запись:

```
child = record
  name: string; {имя и фамилия}
  sum: integer; {суммарное количество баллов}
end;
```

а для хранения информации обо всех учениках — массив записей. Как следует из текста задачи, максимальное число учеников — 100. Следовательно, число элементов массива также будет равно 100:

```
p: array [1..100] of child;
```

Алгоритм работы программы:

1) считывание количества учеников, сдавших экзамен:

```
readln(N);
```

2) заполнение массива записей данными, вводимыми с клавиатуры:

2.1) чтение и сохранение фамилии ученика — используется цикл **repeat**, в котором считывается очередной символ введенной строки и дописывается в конец поля *name* записи, связанной с *i*-м учеником:

```
repeat
  read(c);
  p[i].name := p[i].name + c;
until c = ' ';
```

Данная программная конструкция используется для чтения как фамилии, так и имени ученика;

2.2) вычисление количества баллов, набранных учеником:

```
readln(b1, b2, b3); {чтение баллов}
p[i].sum := b1 + b2 + b3;
```

3) определение трех худших учеников. Для вычисления баллов трех худших учеников введем три переменные: $s1$, $s2$ и $s3$. Для значений этих переменных должно выполняться условие: $s1 \leq s2 \leq s3$.

Предположим, что были введены данные о четырех учениках, после чего для каждого из них было вычислено суммарное количество баллов:

<code>p[1].name</code>	'Иванов Сергей'
<code>p[1].sum</code>	10
<code>p[2].name</code>	'Алексеев Петр'
<code>p[2].sum</code>	11
<code>p[3].name</code>	'Зими́на Настя'
<code>p[3].sum</code>	12
<code>p[4].name</code>	'Мозжухина Аня'
<code>p[4].sum</code>	5

Каждый ученик может набрать максимально 15 баллов. Поэтому мы инициализируем переменные $s1$, $s2$ и $s3$ значениями, превышающими 15:

```
s1 := 16;
s2 := 16;
s3 := 16;
```

После этого в цикле (в данном случае количество итераций цикла равно 4, т. е. числу учеников) выполняется поиск трех худших учеников:

1) $i = 1$; так как `p[i].sum < s1`, выполняется обновление значений переменных $s1$, $s2$ и $s3$:

```
s3 := s2;
s2 := s1;
s1 := p[i].sum; {в s1 записываем новое значение}
```

Получаем: $s1 = 10$, $s2 = 16$, $s3 = 16$;

2) $i = 2$; так как `p[i].sum` не меньше $s1$, проверяется условие: `p[i].sum < s2`; данное условие выполняется, поэтому обновляются значения переменных $s2$ и $s3$:

```
s3 := s2;
s2 := p[i].sum;
```

Получаем: $s1 = 10$, $s2 = 11$, $s3 = 16$;

3) $i = 3$; `p[i].sum` не меньше $s1$; `p[i].sum` не меньше $s2$; проверяется условие: `p[i].sum < s3`; данное условие выполняется, поэтому вычисляется новое значение переменной $s3$:

```
s3 := p[i].sum;
```


Получаем: $s1 = 10, s2 = 11, s3 = 12$;

4) $i = 4$; так как $p[i].sum < s1$, обновляется значение переменных $s1, s2$ и $s3$:

```
s3 := s2;  
s2 := s1;  
s1 := p[i].sum; {в s1 записываем новое значение}  
Получаем:  $s1 = 5, s2 = 10, s3 = 11$ .
```

После завершения работы этого алгоритма в $s3$ хранится наибольшее количество баллов, которое набрали худшие ученики. Поэтому для вывода всех худших учеников нужно вывести тех учеников, у которых количество баллов меньше или равно $s3$:

```
for i := 1 to N do  
  if p[i].sum <= s3 then  
    writeln(p[i].name);
```

Пример правильной и эффективной программы на языке Паскаль, решающей данную задачу:

```
program SchoolTest;  
type  
  child = record  
    name: string; {имя и фамилия}  
    sum: integer; {суммарное количество баллов}  
  end;  
var  
  p: array[1..100] of child;  
  c: char;  
  i, j, N, s1, s2, s3, b1, b2, b3: integer;  
begin  
  readln(N);  
  for i := 1 to N do  
    begin  
      p[i].name := '';  
      for j := 1 to 2 do  
        repeat  
          read(c);  
          p[i].name := p[i].name + c;  
        until c = ' '; {чтение имени и фамилии}  
      readln(b1, b2, b3); {чтение баллов}  
      p[i].sum := b1 + b2 + b3;  
    end;
```

```
s1 := 16;
s2 := 16;
s3 := 16;
for i := 1 to N do
begin
  if p[i].sum < s1 then
  begin
    s3 := s2;
    s2 := s1;
    s1 := p[i].sum;
  end
  else
  if p[i].sum < s2 then
  begin
    s3 := s2;
    s2 := p[i].sum;
  end
  else
  if p[i].sum < s3 then
    s3 := p[i].sum;
  end;
for i := 1 to N do
  if p[i].sum <= s3 then
    writeln(p[i].name);
readln;
end.
```

5.11.5. Тестирование программ

Цель тестирования программы заключается в оценке правильности обработки ею введенных пользователем данных и в проверке совпадения получаемых результатов работы с ожидаемыми результатами.

Чтобы выполнить тестирование программы **SchoolTest**, необходимо ввести N строк с клавиатуры, причем в случае неверного ввода нужно выполнять процесс ввода данных заново. Конечно, при небольшом N можно ввести данные с клавиатуры, однако при больших значениях N вводить данные таким способом достаточно утомительно.

Одной из возможностей упростить ввод данных является использование *оператора перенаправления ввода*. При его использовании

данные, отправляемые на вход программе, можно заранее сохранить в текстовый файл, а затем передать их на вход программы. Оператор перенаправления ввода обозначается знаком «<».

Пример записи команды с перенаправлением ввода:

```
SchoolTest.exe < DataFile.txt
```

Содержимое файла `DataFile.txt` при этом, например, для ранее рассмотренной задачи с номером 3 может быть следующим:

```
10
Иванов Петр 3 4 3
Демин Дмитрий 3 5 5
Сухова Светлана 3 3 3
Петрова Лидия 4 4 5
Засухин Иван 3 3 5
Канаев Сергей 3 3 3
Шитов Александр 3 4 3
Синицина Вера 3 5 5
Торгашов Василий 5 5 5
Соколова Евгения 5 5 5
```

Задача 4. На вход программе подаются сведения о сдаче экзаменов абитуриентами, поступающими на одну и ту же специальность. В первой строке сообщается количество абитуриентов N , которое не превосходит 100, а каждая из следующих N строк имеет формат:

<Фамилия> <Инициалы> <баллы> <льгота> ,

где <Фамилия> — строка, состоящая не более чем из 20 символов, <Инициалы> — строка, состоящая из 4 символов, <баллы> — записанные через пробел три целых числа, соответствующие оценкам по стобалльной системе, а <льгота> — число 0 или 1 («нет льгот» или «льгота есть»). Записи <Фамилия> и <Инициалы>, <Инициалы> и <баллы>, а также <баллы> и <льгота> разделены одним пробелом.

Пример входной строки:

```
Иванов П.К. 45 57 38 0
```

По плану в учебную группу должно быть зачислено 15 абитуриентов. Их зачисление проводится по следующим правилам:

1) в первую очередь зачисляются абитуриенты, имеющие льготы, если они набрали более 30 баллов по каждому предмету;

2) далее зачисляются абитуриенты в порядке убывания суммы баллов по трем предметам. Абитуриент, не имеющий льгот, рекомендуется к зачислению, если он набрал по каждому предмету более 40 баллов.

Требуется написать программу, которая будет выводить на экран одно из сообщений о результатах зачисления:

1) «список успешно сформирован» — если количество льготников и абитуриентов, набравших необходимое количество баллов, равно 15 человек;

2) «имеются вакантные места» — если количество льготников и абитуриентов, набравших необходимое количество баллов, меньше 15 человек;

3) «к зачислению рекомендуется больше 15 абитуриентов» — если количество льготников и абитуриентов, набравших необходимое количество баллов, больше 15 человек.

Решение.

Информацию о фамилии и инициалах абитуриентов в программе анализировать не требуется, поэтому при считывании данных можно их не сохранять. Введем следующие переменные:

1) *lgt* — количество льготников, набравших по каждому предмету более 30 баллов;

2) *abt* — количество абитуриентов, не имеющих льгот и набравших по каждому предмету более 40 баллов, а в сумме — не менее 120 баллов.

Алгоритм работы программы:

1) считывается количество абитуриентов, сдававших экзамены;

2) считывается количество баллов, набранных каждым абитуриентом, и информация о льготе;

3) анализируется значение переменной *l1*, в которой хранится информация о том, имеет ли данный абитуриент льготу:

3.1) если абитуриент имеет льготу (*l1* = 1) и набрал по каждому предмету более 30 баллов, то значение переменной *lgt* увеличивается на 1;

3.2) если абитуриент не имеет льготы (*l1* = 0) и набрал по каждому предмету более 40 баллов, то значение переменной *abt* увеличивается на 1;

4) после считывания всех строк в зависимости от значения переменных *lgt* и *abt* выводятся сообщения, указанные в тексте задания:

```
if (lgt + abt) = 15 then
    writeln('список успешно сформирован');
if (lgt + abt) < 15 then
    writeln('имеются вакантные места');
if (lgt + abt) > 15 then
    writeln('к зачислению рекомендуется '
        'больше 15 абитуриентов');
```

Решение задачи приведено в программе **Abitur1**:

```
program Abitur1;
var
  c: char;
  abt, lgt, i, l1, b1, b2, b3, N: integer;
begin
  lgt := 0;
  abt := 0;
  readln(N);
  for i := 1 to N do
    begin
      repeat
        read(c);
      until c = ' '; {считана фамилия}
      repeat
        read(c);
      until c = ' '; {считаны инициалы}
      readln(b1, b2, b3, l1); {считаны баллы и льгота}
      if (l1 = 1) then
        if (b1 > 30) and (b2 > 30) and (b3 > 30) then
          lgt := lgt + 1
        else
          if (b1 > 40) and (b2 > 40) and (b3 > 40) then
            abt := abt + 1;
      end;
      if (lgt + abt) = 15 then
        writeln('список успешно сформирован');
      if (lgt + abt) < 15 then
        writeln('имеются вакантные места');
      if (lgt + abt) > 15 then
        writeln('к зачислению рекомендуется '
          'больше 15 абитуриентов');
      readln;
    end.
end.
```

Задача 5. На вход программы подаются сведения о сдаче экзаменов абитуриентами, поступающими на одну и ту же специальность. В первой строке сообщается количество абитуриентов N , которое не превосходит 100, а каждая из следующих N строк имеет формат:

<Фамилия> <Инициалы> <баллы> <льгота>,

где <Фамилия> — строка, состоящая не более чем из 20 символов, <Инициалы> — строка, состоящая из 4 символов, <баллы> — записанные через пробел три целых числа, соответствующие оценкам по стобальной системе, а <льгота> — число 0 или 1 («нет льгот» или «льгота есть»). Записи <Фамилия> и <Инициалы>, <Инициалы> и <баллы>, а также <баллы> и <льгота> разделены одним пробелом.

Пример входной строки:

Иванов П.К. 45 57 38 0

По плану должно быть зачислено 15 абитуриентов. Зачисление абитуриентов проводится по следующим правилам:

- 1) в первую очередь зачисляются абитуриенты, имеющие льготы, если они набрали более 30 баллов по каждому предмету;
- 2) далее зачисляются абитуриенты в порядке убывания суммы баллов по трем предметам.

Требуется написать программу, которая будет выводить на экран фамилии и инициалы рекомендованных к зачислению абитуриентов. (Предполагается, что все абитуриенты набрали в сумме разное количество баллов.)

Решение.

Алгоритм работы программы:

- 1) считывается количество абитуриентов, сдававших экзамены;
- 2) считывается количество баллов, набранных каждым абитуриентом, и информация о льготе;
- 3) анализируется значение переменной *l1*, в которой хранится информация о том, имеет ли данный абитуриент льготу:
 - 3.1) если абитуриент имеет льготу (*l1* = 1) и набрал по каждому предмету более 30 баллов, то значение переменной *lgt* увеличивается на единицу, а на экран выводится фамилия и имя такого абитуриента;
 - 3.2) если абитуриент не имеет льготы (*l1* = 0), то информация об этом абитуриенте сохраняется в массиве:


```
p[abt].name := str;
p[abt].sum := b1 + b2 + b3;
abt := abt + 1;
```
- 4) после считывания всех строк анализируется значение переменной *lgt*. Если оно меньше 15, то это означает, что из общей квоты в 15 мест существуют места, еще не занятые льготниками. В этом случае выполняется сортировка массива абитуриентов, не имеющих льгот, по убыванию числа набранных баллов, и фамилии и имена таких абитуриентов выводятся на экран:

```
if (lgt < 15) then
begin
  for i := 1 to N2-1 do
    for j := 1 to N2-1 do
      if p[j].sum < p[j+1].sum then
      begin
        t := p[j];
        p[j] := p[j+1];
        p[j+1] := t;
      end;
    for i := 1 to 15-lgt do
      writeln(p[i].name);
```

Решение задачи показано в программе **Abitur2**:

```
Program Abitur2;
type
  child = record
    name: string; {имя и фамилия}
    sum: integer; {суммарное количество баллов}
  end;
var
  p: array [1..100] of child;
  t: child;
  c: char;
  abt, lgt, i, j, l1, b1, b2, b3, N, N2, k: integer;
  str: string;
begin
  lgt := 0;
  abt := 1;
  readln(N);
  for i := 1 to N do
    begin
      str := '';
      for k := 1 to 2 do
        repeat
          read(c);
          str := str + c;
        until c = ' '; {считана фамилия и имя}
      readln(b1, b2, b3, l1); {считаны баллы и льгота}
      if (l1 = 1) then
```

```

begin
  if (b1 > 30) and (b2 > 30) and (b3 > 30)
  then
    begin
      lgt := lgt + 1;
      writeln(str);
    end
  end
else
  begin
    p[abt].name := str;
    p[abt].sum := b1 + b2 + b3;
    abt := abt + 1;
  end;
end;
N2 := abt - 1;
if (lgt < 15) then
begin
  for i := 1 to N2-1 do
    for j := 1 to N2-1 do
      if p[j].sum < p[j+1].sum then
        begin
          t := p[j];
          p[j] := p[j+1];
          p[j+1] := t;
        end;
      for i := 1 to 15-lgt do
        writeln(p[i].name);
      end;
    readln;
  end.

```

Задача 6. На вход программы подаются сведения о номерах школ учащихся, участвовавших в олимпиаде. В первой строке сообщается количество учащихся N , а каждая из следующих N строк имеет формат:

<Фамилия> <Инициалы> <номер школы>,

где <Фамилия> — строка, состоящая не более чем из 20 символов, <Инициалы> — строка, состоящая из 4 символов (буква, точка, буква, точка), а <номер школы> — не более чем двузначный номер. Записи <Фамилия> и <Инициалы>, а также <Инициалы> и <номер школы> разделены одним пробелом.

Пример входной строки:

Иванов П.С. 57

Требуется написать программу, которая будет выводить на экран информацию, из каких школ (от которых был хотя бы один участник) было меньше всего участников олимпиады [6].

Решение.

Информацию о фамилии и инициалах учащихся в программе анализировать не требуется, поэтому при считывании данных эта информация не сохраняется. Введем следующие переменные:

1) `schools: array [1..99] of integer`; — массив, значение i -го элемента которого соответствует количеству учеников, учащихся в i -й школе;

2) `num` — переменная, в которую записывается номер школы для данного ученика.

Алгоритм работы программы:

1) считывается количество школьников, участвовавших в олимпиаде;

2) производится заполнение массива `schools` данными, вводимыми с клавиатуры:

2.1) пропуск фамилии и имени ученика:

```
for k := 1 to 2 do
  repeat
    read(c);
  until c = ' '; {пропуск фамилии и имени}
```

2.2) чтение и сохранение номера школы ученика:

```
readln(num); {чтение номера школы}
schools[num] := schools[num] + 1;
```

3) выполняется поиск минимального элемента массива `schools`:

```
min := N;
for i := 1 to 99 do
  if (schools[i] > 0) and (schools[i] < min) then
    min := schools[i];
```

Решение задачи показано в программе **SchoolTest2**:

```
program SchoolTest2;
```

```
var
```

```
  schools: array [1..99] of integer;
```

```
  num: integer;
```

```
  c: char;
```

```
  i, k, N, min: integer;
```

```

begin
  readln(N);
  for i := 0 to 99 do
    schools[i] := 0;
  for i := 1 to N do
    begin
      for k := 1 to 2 do
        repeat
          read(c);
          until c = ' '; {пропуск фамилии и имени}
          readln(num); {чтение номера школы}
          schools[num] := schools[num] + 1;
        end;
      min := N;
      for i := 1 to 99 do
        if (schools[i] > 0) and (schools[i] < min) then
          min := schools[i];
      for i := 1 to 99 do
        if schools[i] = min then
          writeln(i);
      readln;
    end.

```

5.12. Множества

Множество представляет собой набор однотипных объектов, при этом, в отличие от массива, порядок следования элементов в множестве не важен. Тип элементов множества может быть перечисляемым или интервальным.

Определение переменной — множества:

```

var
  letters: set of 'a' .. 'z';

```

Тип-множество определяется следующим образом:

```

type
  digitChar = set of '0' .. '9';
var
  d1, d2, d3, d4: digitChar;
  lgc: boolean;

```

Количество элементов множества может изменяться от 0 до 255. Множество, не содержащее элементов, называется *пустым* и обозначается как [].

Операции, применимые к величинам типа множество:

Операция	Запись	Результат	Примечание
<code>:=</code>	<code>d1 := ['1', '2', '3']; d2 := ['3', '4', '5']; d3 := ['1'..'9']; d4 := ['1', '5'];</code>	<code>d4 = ['1', '5']</code>	Инициализация переменной типа множество
<code>*</code> (пересечение множеств)	<code>d3 := d1 * d2;</code>	<code>d3 = ['3']</code>	Результат содержит общие для множеств <i>d1</i> и <i>d2</i> элементы
<code>+</code> (объединение множеств)	<code>d3 := d1 + d2;</code>	<code>d3 = ['1', '2', '3', '4', '5']</code>	Результат содержит элементы множества <i>d1</i> , дополненные недостающими элементами множества <i>d2</i>
<code>-</code> (разность множеств)	<code>d3 := d1 - d2;</code>	<code>d3 = ['1', '2']</code>	Результат содержит элементы из множества <i>d1</i> , которые не принадлежат множеству <i>d2</i>
<code>=</code> (эквивалентность множеств)	<code>lgc := d1 = d2;</code>	<code>lgc = false</code>	Результат <code>true</code> , если множества <i>d1</i> и <i>d2</i> эквивалентны, и <code>false</code> — в противном случае
<code><></code> (не эквивалентность множеств)	<code>lgc := d1 <> d2;</code>	<code>lgc = true</code>	Результат <code>true</code> , если множества <i>d1</i> и <i>d2</i> не эквивалентны, и <code>false</code> — в противном случае
<code><=</code> («содержится в»)	<code>lgc := d1 <= d2;</code>	<code>lgc = true</code>	Результат <code>true</code> , если множество <i>d1</i> содержится в множестве <i>d2</i>

Операция	Запись	Результат	Примечание
<code>>=</code> («содержит»)	<code>lgc := d1 >= d3;</code>	<code>lgc = false</code>	Результат <code>true</code> , если множество <i>d1</i> содержит в себе множество <i>d3</i>
<code>in</code> (проверка принадлежности)	<code>lgc := '1' in d3;</code>	<code>lgc = true</code>	Результат <code>true</code> , если множество <i>d3</i> содержит элемент '1'
<code>include(s, i)</code> (включение элемента)	<code>include(d1, '4');</code>	<code>d1 = ['1', '2', '3', '4']</code>	Включает элемент <i>i</i> в множество <i>s</i>
<code>exclude(s, i)</code> (исключение элемента)	<code>exclude(d1, '3')</code>	<code>d1 = ['1', '2']</code>	Исключает элемент <i>i</i> из множества <i>s</i>

Инициализация величин типа множество может быть выполнена с помощью типизированных констант:

```
const
  seLit: set of 'A'..'D' = [];
```

5.13. Работа с файлами

Алгоритм работы с файлом обычно включает в себя следующие шаги:

- 1) объявление *файловой переменной*;
- 2) связывание файловой переменной с файлом;
- 3) открытие файла для чтения или записи;
- 4) выполнение над файлом определенных операций;
- 5) закрытие файла.

Процедуры и функции компилятора FreePascal позволяют работать с тремя видами файлов — текстовыми, типизированными и не-типизированными.

Текстовый файл представляет собой набор строк, где каждая строка завершается *признаком конца строки* и состоит из символов таблицы ASCII. Длина строк в файле может быть различной. В текстовом файле может храниться любая информация, но при ее обработке она будет всегда рассматриваться как текстовая.

Типизированный файл предназначен для хранения значений какого-либо одного типа (в этом он похож на массив).

Нетипизированный файл может содержать *любую* информацию и представляет собой набор блоков размером n байт ($n \in [1; 65535]$). Как значение n , так и содержимое блоков определяется программистом, причем содержимое разных блоков может различаться.

Переменная, связываемая с файлом, объявляется следующим образом:

- для текстового файла:
var
 TextFile: Text;
- для типизированного файла:
var
 TypedFile: <тип>;
- для нетипизированного файла:
var
 UnTypedFile: File

После того как файловая переменная объявлена, необходимо связать ее с файлом, используя процедуру `assign`:

```
assign(var f: <тип файла>; const Name: string);
```

где f — файловая переменная, а $Name$ — константа или переменная, содержащая имя и путь к открываемому файлу.

Открытие файла для чтения или записи информации можно выполнить с помощью различных процедур:

1) процедура `reset`:

- текстовый файл:
`reset(var f: Text);` — файл открывается для чтения;
- типизированный файл:
`reset(var t: <тип>);` — файл открывается для чтения и записи;
- нетипизированный файл:
`reset(var t: File [; L: LongInt]);`
где L — размер записи файла, — файл открывается для чтения и записи.

При использовании процедуры `reset` открываемый файл должен уже существовать на диске, иначе возникнет ошибка;

2) процедура `rewrite`:

- текстовый файл:
`rewrite(var f: Text);` — файл открывается для записи;
- типизированный файл:
`rewrite(var t: <тип>);` — файл открывается для чтения и записи;

- нетипизированный файл:

```
rewrite(var t: File [; L: LongInt]);
```

где L — размер записи файла, — файл открывается для чтения и записи.

Особенности использования процедуры `rewrite`:

- если файл уже существует, то его содержимое удаляется;
- если файл не существует, то он создается;
- если указать конкретное значение параметра L (например, 10), то из нетипизированного файла с помощью процедур чтения или записи можно будет считать или записать соответствующее число байтов информации (в данном примере 10 байтов), а если значение параметра L не указано, то считается, что размер записи равен 128 байт;

3) процедура `append` (**var** t: Text) открывает файл в режиме записи. Данная процедура может быть применена только к текстовому файлу.

При этом один и тот же файл открыть одновременно несколькими способами нельзя.

5.13.1. Процедуры чтения и записи информации в файл

1) процедура `read` (либо `readln`) используется для чтения данных из текстового или типизированного файла. Формат записи:

```
read(var f: <тип файла>; args: arguments);
```

где `args: arguments` — список переменных (аргументов), в которые выполняется запись информации, считанной из файла. Если в файле нет данных, то сгенерируется ошибка. Если файл текстовый, то переменные (аргументы) могут относиться к одному из следующих типов: `char`, `integer`, `single` или `string`. Если файл типизированный, то переменные должны быть того же типа, что и тип, указанный в определении файловой переменной.

Процедура `readln` после чтения данных из строки файла выполняет перевод указателя на следующую строку файла. Данная процедура не может быть использована с нетипизированным файлом;

2) процедура `BlockRead` предназначена для чтения данных из нетипизированных файлов:

```
procedure BlockRead(var f: file; var buf; count:
                    LongInt; var result: LongInt);
```

Процедура `BlockRead` выполняет чтение `count` записей из файла f . Здесь под записью понимается блок байтов, размер которого определен с помощью процедуры `rewrite` или `reset`. (Размер за-

писи определяется при открытии файла в параметрах процедуры `reset` или `rewrite`.) Данные, считанные из файла, помещаются в буфер *buf*, размер которого должен соответствовать размеру считываемых данных.

Предположим, что размер записи файла — 3 байта, тогда если в *count* хранится значение 4, то из файла будет считано 12 байтов.

Параметр *result* является необязательным. Если он не определен и считано количество записей, меньшее, чем *count*, то генерируется ошибка. После выполнения процедуры `BlockRead` в переменной *result* находится количество считанных из файла записей;

3) процедура `write` (либо `writeln`) предназначена для записи данных в типизированный или текстовый файл:

```
write(var f: <тип файла>; args: arguments)
```

где *args: arguments* — список переменных (аргументов), значения которых будут записаны в файл, а *f* — переменная, связанная с типизированным или текстовым файлом.

Если запись выполняется в типизированный файл, то переменные, указанные в списке аргументов, должны быть того же типа, что и компоненты типизированного файла.

В процедуре `write` допустимо использование *форматирования*, такого же, как и при выводе на экран. Вещественные числа по умолчанию записываются в формате с плавающей запятой. Данная процедура не может быть использована с нетипизированным файлом.

Процедура `writeln` работает так же, как и процедура `write`. Отличие же заключается в том, что после записи в файл значений переменных выполняется запись символов перевода строки, а если никакие переменные не указываются, то в файл записывается пустая строка;

4) процедура `BlockWrite` выполняет запись данных в нетипизированный файл:

```
procedure BlockWrite(var f: file; const buf; count:
                    LongInt; var result: LongInt);
```

Эта процедура выполняет запись *count* записей из *buf* в файл *f*. Если данные не могут быть записаны в файл, то генерируется ошибка. Фактическое количество сохраненных записей указывается в переменной *result* (значение этой переменной может быть меньше, чем значение *count*).

После окончания работы с файлом для сохранения всех изменений, выполненных в файле, необходимо использовать процедуру `close`:

```
close(var f: <тип файла>);
```

Данная процедура предназначена для закрытия файла *f*. Процедура `close` может применяться к файлам всех типов. После ее вызова к закрытому файлу уже нельзя применять операции чтения и записи данных. Чтобы снова открыть этот же файл, уже не нужно использовать процедуру `assign` — достаточно использовать процедуры `reset`, `rewrite` или `append`.

5.13.2. Дополнительные процедуры и функции для работы с файлами

1. Процедура `rename` изменяет имя файла, связанного с файловой переменной *f* (файл при этом не должен быть открыт!) на указанное новое имя:

```
rename(var f: <тип файла>; const s: string);
```

Данная процедура может применяться к файлам всех типов.

Пример использования процедуры `rename`:

```
program RenameFile;  
var  
    f: Text;  
begin  
    assign(f, 'test.txt');  
    rewrite(f);  
    writeln(f, 'текстовый файл');  
    close(f);  
    rename(f, 'RenamedFile.txt');  
end.
```

2. Процедура `erase` удаляет файл с диска:

```
erase(var f: <тип файла>);
```

Эта процедура может применяться к файлам любых типов. В параметре процедуры `erase` указывается соответствующая файловая переменная. Удаленный файл не должен быть открыт с помощью процедур `assign`, `append` или `rewrite`, иначе будет сгенерирована ошибка.

Пример использования процедуры `erase`:

```
program EraseExample;  
var f: text;  
begin  
    assign(a, 'test.txt');  
    rewrite(a);  
    writeln(a, 'текстовый файл');  
    close(a);  
    erase(a);  
end.
```


3. Функция `FileSize` возвращает количество записей в файле:

```
FileSize(var f: <тип файла>): LongInt;
```

Эта функция может использоваться с типизированными и нетипизированными файлами, но не используется для текстовых файлов. Если файл пуст, то функция `FileSize` вернет значение 0.

4. Процедура `Truncate` удаляет все данные в файле, находящиеся после текущей позиции указателя; она может быть применена только к типизированным и нетипизированным файлам:

```
truncate(var f: <тип файла>);
```

Пример использования процедуры `Truncate`:

```
program TruncateTest;
var
  f: file of Byte;
  i, L: Byte;
begin
  assign(f, 'TestFile.tmp');
  rewrite(f);
  for i := 1 to 5 do
    write(f, i);
  writeln('Размер файла до усечения: ', FileSize(f));
  close(f);
  reset(f);
  repeat
    read(f, i);
  until i = 2;
  truncate(f);
  writeln('Размер файла после усечения: ', FileSize(f));
  close(f);
  readln;
end.
```

5. Процедура `seek` устанавливает указатель в файле на запись с определенным номером. Она может быть применена к типизированным и нетипизированным файлам, но не к текстовым:

```
seek(var f: <тип файла>; Pos: Int64);
```

Самая первая запись в файле имеет номер 0. Если для нетипизированного файла при вызове процедуры `reset` или `rewrite` не указан размер записи, то по умолчанию он равен 128.

6. Функция `eof` возвращает значение `true`, если файловый указатель достиг конца файла или если файл пуст. Данная функция может использоваться для всех видов файлов:

```
eof(var f: <тип файла>): boolean;
```

7. Функция `SeekEof` возвращает значение `true`, если файловый указатель достиг конца файла. Данная функция может быть использована только для текстовых файлов. Ее отличие от функции `eof` заключается в том, что функция `SeekEof` возвращает значение `true`, если между текущим символом и символом конца файла находятся только пробельные символы. Конец файла обозначается специальным символом с кодом #26:

```
SeekEof(var f: Text): boolean
```

Если в текстовом файле встретится символ конца файла (код #26), то функция `SeekEof` вернет значение `true` и завершит свою работу.

8. Функция `EOLn` возвращает значение `true`, если достигнут конец файла; в любых других случаях возвращается значение `false`. Конец строки обозначается специальным символом с кодом #10. Данная функция может применяться только к текстовому файлу:

```
EOLn(var f: Text): boolean;
```

9. Функция `SeekEOLN` возвращает значение `true`, если между указателем и концом строки находятся только пробельные символы. Данная функция может использоваться только с текстовыми файлами:

```
SeekEOLn(var f: Text): boolean;
```

10. Функция `FilePos` возвращает номер текущей записи файла, указанного в ее параметре. Данная функция может использоваться только с типизированными и нетипизированными файлами, но не с текстовыми (так как с текстовым файлом не связано понятие «запись»). Формат записи функции `FilePos`:

```
function FilePos(var f: File): Int64;
```

11. Функция `IOResult` возвращает результат последней операции ввода/вывода, если используется директива компилятора* `{SI-}`. При успешном завершении этой операции данная функция возвращает значение 0, иначе — код ошибки (положительное число):

```
IOResult: word;
```

* Директива компилятора — команда, которая позволяет управлять процессом компиляции программы.

Задача 1. На автозаправочных станциях (АЗС) продается бензин с маркировкой 92, 95 и 98. В городе N был проведен мониторинг цены бензина на различных АЗС. Напишите программу, которая будет определять для каждого вида бензина, сколько АЗС продают его дешевле всего.

Данные находятся в текстовом файле, в первой строке которого находится количество данных о стоимости бензина, а в каждой из последующих N строк находится информация в следующем формате:

<Компания> <Улица> <Марка> <Цена> ,

где <Компания> — строка, состоящая не более чем из 20 символов без пробелов, <Улица> — строка, состоящая не более чем из 20 символов без пробелов, <Марка> — одно из чисел: 92, 95 или 98, а <Цена> — целое число в диапазоне от 1000 до 3000, обозначающее стоимость одного литра бензина в копейках. Записи <Компания> и <Улица>, <Улица> и <Марка>, а также <Марка> и <Цена> разделены ровно одним пробелом. Пример строки файла:

Синойл Цветочная 95 22.50

Программа должна выводить через пробел три числа — количество АЗС, продающих дешевле всего 92-й, 95-й и 98-й бензин соответственно. Если бензин какой-то марки нигде не продавался, то следует вывести 0. Пример выходных данных:

12 1 0

Напишите программу для генерации содержимого файла, формат которого описан выше.

Решение.

Нам необходимо хранить информацию о трех марках бензина. Для этого можно использовать одномерный массив с диапазоном индексов от 92 до 98 (где из 7 элементов будут использоваться три).

Для определения количества АЗС, продающих бензин дешевле всего, нам не требуется выполнять анализ полей <Компания> и <Улица>, находящихся в начале каждой строки. Данные, находящиеся в этих двух полях, разделены пробелами, поэтому, чтобы пропустить содержимое этих полей, нужно считывать символы до тех пор, пока не будут считаны два пробела. Программная реализация пропуска этих символов выглядит следующим образом:

```
for k := 1 to 2 do
  repeat
    read(F, c);
  until c = ' ';
```

Здесь в цикле `repeat` символы из файла считываются до тех пор, пока не встретится пробел. А чтобы цикл `repeat` выполнялся дважды (ведь нам необходимо пропустить два пробела), он включен в цикл `for`.

После того как первые два поля пропущены, считываются значения полей, которые относятся к марке бензина и его цене:

```
readln(F, k, b);
```

Далее возможны следующие варианты поведения программы:

1) если считанное значение цены бензина меньше того, которое хранится в i -м элементе массива *min* для соответствующей марки бензина, то i -е значение массива заменяется на минимальное, а в массиве *cnt* i -й элемент устанавливается в 1;

2) если считанное значение цены бензина равно значению, хранящемуся в i -м элементе массива *min*, то количество АЗС, установивших указанную цену, увеличивается на 1:

```
if min[k] > b then
begin
    min[k] := b;
    cnt[k] := 1
end
else
    if min[k] = b then
        cnt[k] := cnt[k] + 1;
```

Программа, решающая первую часть задачи (выполняющая анализ цен на бензин):

```
program AnalyseTextFile;
var
    min, cnt: array [92..98] of integer;
    c: char;
    i, k, N, b: integer;
    F: text;
begin
    assign(F, 'TextData.txt');
    reset(F);
    for i := 92 to 98 do
        begin
            min[i] := 3001;
            cnt[i] := 0;
        end;
    readln(F, N);
```

```
for i := 1 to N do
begin
  for k := 1 to 2 do
  repeat
    read(F, c);
  until c = ' ';
  readln(F, k, b);
  if min[k] > b then
  begin
    min[k] := b;
    cnt[k] := 1
  end
  else
  if min[k] = b then
    cnt[k] := cnt[k] + 1;
  end;
writeln(cnt[92], ' ', cnt[95], ' ', cnt[98]);
close(F);
readln;
end.
```

Программа, решающая вторую часть задачи (генерирующая файл с исходными данными):

```
program GenTextFile;
var
  F: text;
  N, M, i: byte;
  cost: word;
begin
  assign(F, 'TextData.txt');
  rewrite(F);
  write('число генер. записей = ');
  readln(N);
  writeln(F, N);
  for i := 1 to N do
  begin
    write(F, 'Comp', i, ' Strt', i);
    N := trunc(random(3));
    case N of
      0: write(F, ' ', 92);
      1: write(F, ' ', 95);
      2: write(F, ' ', 98);
    end;
  end;
```

```

        cost := 10 + trunc(random(10));
        writeln(F, ' ', cost);
    end;
    close(F);
end.

```

Задача 2. Разработать программу, которая генерирует типизированный файл, содержащий вещественные числа. Разработать программу, которая считывает числа из сгенерированного файла и выполняет поиск максимального числа.

Решение.

Программа **GenTypedFile** выполняет генерацию файла, а программа **AnalyseTypedFile** осуществляет чтение и анализ его содержимого.

```

program GenTypedFile;
var
    F: file of single;
    R: single;
    N, i: word;
begin
    randomize;
    assign(F, 'FileOfSingle.txt');
    rewrite(F);
    write('Введите количество чисел = ');
    readln(N);
    for i := 1 to N do
        begin
            R := random;
            write(F, R);
        end;
    close(F);
end.

program AnalyseTypedFile;
var
    F: file of single;
    R, max: single;
    N, i: word;
begin
    assign(F, 'FileOfSingle.txt');
    reset(F);
    N := FileSize(F);
    max := -1;

```

```
for i := 1 to N do
begin
  read(F, R);
  if R > max then
    max := R;
end;
writeln('Макс. число = ', max:2:3);
close(F);
readln;
end.
```

Для определения количества записей в файле используется функция `FileSize`.

Задача 3. Разработать программу, генерирующую типизированный файл, элементами которого являются массивы целых чисел. Разработать программу, которая считывает массивы из сгенерированного файла и выполняет поиск максимума в каждом массиве.

Решение.

Программа **WriteArrays** выполняет запись двух целочисленных массивов в типизированный файл. Для этого используется процедура `write`. Запись файла — целочисленный массив, состоящий из 10 элементов.

Программа **ReadArrays** выполняет чтение данных из файла **arrays.tpf** с помощью цикла **while**, на каждой итерации которого:

- 1) считывается очередная запись (массив из 10 элементов);
- 2) файловый указатель при вызове процедуры `read` смещается на одну запись;
- 3) выполняется поиск максимального элемента массива.

Выполнение цикла прекращается, когда файловый указатель достигнет конца файла. Эту проверку выполняет функция `eof`.

```
Program WriteArrays;
const
  N = 10;
type
  t = array [1..10] of byte;
  v = file of t;
var
  a1, a2: t;
  f: v;
  i: integer;
```

```
begin
  assign(f, 'arrays.tpf');
  for i := 1 to N do
    begin
      a1[i] := random(20);
      a2[i] := random(100);
    end;
  rewrite(f);
  write(f, a1);
  write(f, a2);
  close(f);
end.

Program ReadArrays;
type
  t = array [1..10] of byte;
  v = file of t;
var
  a: t;
  f: v;
  r: byte;
  i: integer;
begin
  assign(f, 'arrays.tpf');
  reset(f);
  writeln('Число записей = ', FileSize(f));
  while not eof(f) do
    begin read(f, a);
      r := a[1];
      for i := 2 to 10 do
        if a[i] > r then
          r := a[i];
        writeln('max = ', r);
      end;
    close(f);
    readln;
  end.
```

Задача 4. Разработать программу, выполняющую запись в файл следующих данных: строки, вещественного числа и записи, состоящей из трех полей. Разработать программу, выполняющую чтение из файла описанных выше данных и выводящую их на экран.

Решение.

В программе **GenUnTypedFile** показаны два варианта использования процедуры **BlockWrite**:

1) с учетом фактического количества сохраненных в файл записей:

```
BlockWrite(F, SomeRec, RecSz, Res);
```

2) без учета количества сохраненных в файл записей:

```
BlockWrite(F, S, SizeOf(S));
```

Если фактическое количество сохраненных записей (*Res*) и количество сохраненных записей (*RecSz*) не равны друг другу, то выводится сообщение об ошибке, а выполнение программы завершается:

```
if Res <> RecSz then
begin
  writeln('Ошибка записи');
  readln;
  close(F);
  halt;
end;
```

Функция **SizeOf(X) : word** возвращает объем (в байтах), занимаемый величиной *X* в памяти. Функция **halt** прерывает выполнение программы.

Программа **GenUnTypedFile** выполняет запись, а программа **AnalyseUnTypedFile** — чтение описанных в условии задачи данных из нетипизированного файла.

```
program GenUnTypedFile;
const x = 2.5;
var
  F: file;
  R: single;
  S: string[6];
  SomeRec: record
    Field1: char;
    Field2: single;
  end;
  Res, RecSz: word;
begin
  RecSz := SizeOf(SomeRec);
  assign(F, 'UnTypedFile.utp');
  rewrite(F, 1);
  R := 2.7;
```

```
S := 'строка';
SomeRec.Field1 := 'x';
SomeRec.Field2 := 0.1;
BlockWrite(F, R, SizeOf(R));
BlockWrite(F, S, SizeOf(S));
BlockWrite(F, SomeRec, RecSz, Res);
if Res <> RecSz then
    begin
        writeln('Ошибка записи');
        readln;
        close(F);
        halt;
    end;
close(F);
readln;
end.

program AnalyseUnTypedFile;
const x = 2.5;
var
    F: file;
    R: single;
    S: string[6];
    SomeRec: record
        Field1: char;
        Field2: single;
    end;
    Res, RecSz: word;
begin
    RecSz := SizeOf(SomeRec);
    assign(F, 'UnTypedFile.utp');
    reset(F, 1);
    BlockRead(F, R, SizeOf(R));
    Writeln(R:2:3);
    BlockRead(F, S, SizeOf(S));
    Writeln(S);
    BlockRead(F, SomeRec, RecSz, Res);
    Writeln(SomeRec.Field1);
    Writeln(SomeRec.Field2:2:3);
    if Res <> RecSz then
        begin
            writeln('Ошибка чтения');
            readln;
```

```
        close(F);  
        halt;  
    end;  
    close(F);  
    readln;  
end.
```

5.13.3. Процедуры для работы с каталогами

Для работы с каталогами предназначены следующие процедуры:

- 1) `chdir(s: string)` — делает текущим каталог, указанный в строке *s*. Если такого каталога нет, то произойдет ошибка;
- 2) `getdir(disk: byte; s: string)` — записывает в строку *s* имя текущего каталога на указанном диске (0 — текущий диск, 1 — диск **A**, 2 — диск **B** и т. д.);
- 3) `mkdir(s: string)` — создает в текущем каталоге подкаталог с указанным в строке *s* именем. Если в текущем каталоге уже существуют каталог с таким именем, то произойдет ошибка;
- 4) `rmdir(s: string)` — удаляет пустой каталог с заданным в строке *s* именем. Если такого каталога нет, то произойдет ошибка.

5.14. Подпрограммы

Подпрограммы являются средством описания фрагментов программы, повторяющихся в разных ее частях.

Подпрограммой называется оформленный фрагмент программы, имеющий собственное имя. Упоминание имени подпрограммы в тексте основной программы приводит к запуску подпрограммы и называется ее *вызовом*. После вызова подпрограммы выполняются входящие в нее операторы. После выполнения последнего оператора подпрограммы начинают выполняться операторы, стоящие за оператором вызова подпрограммы. Обращение к подпрограмме может быть выполнено в любом месте основной программы или другой подпрограммы.

При работе с подпрограммами важными являются понятия формальных и фактических параметров. *Фактические параметры* указываются при вызове подпрограммы в теле основной программы. *Формальные параметры* используются при записи текста подпрограммы. Тип и порядок записи фактических параметров должны быть такими же, как и для формальных параметров. В противном случае результат работы программы будет непредсказуемым.

5.14.1. Описание и вызов подпрограмм

Структура подпрограммы подобна структуре программы. Подпрограмма включает заголовок, раздел описаний и исполняемую часть. причем раздел описаний содержит те же подразделы, что и раздел описаний основной программы, — описание констант, типов, меток, процедур, функций, переменных. Исполняемая часть содержит операторы.

Существуют два типа подпрограмм — *процедуры* и *функции*.

Описание процедуры:

```
procedure <имя_процедуры> (параметры);
[uses <имена_подключаемых_модулей>;]
[label <список_меток>;]
[const <имя_константы> = <значение_константы>;]
[type <имя_типа> = <определение_типа>;]
[var <имя_переменной> : <тип_переменной>;]
[procedure <имя_процедуры>
<описание_процедуры>;]
[function <имя_функции>
<описание_функции>;]
begin
    <операторы>;
end;
```

Вызов процедуры производится следующим образом:

```
<имя_процедуры> (список фактических параметров);
```

Порядок выполнения подпрограммы:

1) фактические параметры подставляются вместо формальных, стоящих на тех же местах в заголовке (т. е. происходит *передача входных параметров*); все формальные параметры создаются в момент вызова подпрограммы и уничтожаются в момент выхода из нее;

2) выполняются операторы исполняемой части процедуры;

3) происходит возврат в вызывающий блок; *передача выходных параметров* происходит непосредственно во время работы исполняемой части.

Описание функции:

```
function <имя_функции> (параметры) : <тип результата>;
uses <имена_подключаемых_модулей>;]
label <список_меток>;]
```

```
const <имя_константы> = <значение_константы>;]  
type <имя_типа> = <определение_типа>;]  
var <имя_переменной> : <тип_переменной>;]  
procedure <имя_процедуры>  
  <описание_процедуры>;]  
function <имя_функции>  
  <описание_функции>;]  
begin  
  <операторы>;  
end;
```

Для передачи в вызывающую программу результата работы функции последним оператором в разделе операторов функции необходимо поместить команду:

```
<имя_функции> := результат;
```

Вызов функции выполняется следующим образом:

```
<результат> := <имя_функции> (список фактических  
                               параметров);
```

Существует также возможность осуществлять вызов функции внутри выражения: имя функции может стоять в правой части оператора присваивания, в разделе условий оператора **if** и т. д.

При вызове процедур и функций необходимо соблюдать следующие правила:

- 1) количество фактических параметров должно совпадать с количеством формальных;

- 2) соответствующие фактические и формальные параметры должны совпадать по порядку следования и по типу.

Имена формальных и фактических параметров могут совпадать. Это не приводит к проблемам, так как соответствующие им переменные будут различными из-за того, что они хранятся в разных областях памяти.

5.14.2. Параметры подпрограмм

Существует три вида параметров:

- 1) *параметры-значения* — локальные переменные подпрограммы, начальные значения которых задаются при вызове этой подпрограммы (им присваиваются значения соответствующих фактических параметров). Параметры-значения, описанные в заголовке подпрограммы, могут изменять свои значения наряду с прочими переменными, но эти изменения строго локальные и никак не от-

ражаются на значениях фактических параметров. Синтаксис описания параметров-значений:

```
<имя_подпрограммы> (P1: <тип_1>; P2: <тип_2>; ... );
```

В качестве фактических параметров подпрограммы, подставляемых на место формальных параметров-значений, могут выступать значения переменных, констант и выражений;

2) при использовании *параметров-переменных* в подпрограмму передается ссылка на ячейку (ячейки) памяти, в которой (которых) хранится значение фактического параметра. При этом все действия внутри подпрограммы с формальным параметром-переменной на самом деле являются действиями над фактическим параметром. Синтаксис описания параметров-переменных:

```
<имя_подпрограммы> (var P1: <тип_1>;  
                    var P2: <тип_2>; ... );
```

Описание параметров-переменных отличается от описания параметров-значений наличием ключевого слова **var** перед идентификатором параметра.

В качестве фактических параметров подпрограммы, подставляемых на место формальных параметров-переменных, могут выступать только переменные;

3) *параметры-константы* аналогичны параметрам-значениям с той лишь разницей, что их значения нельзя изменять в теле подпрограммы. Синтаксис описания параметров-констант:

```
<имя_подпрограммы> (const P1: <тип_1>;  
                    const P2: <тип_2>; ... );
```

Программы **ProcMin** и **FuncMin** демонстрируют использование процедуры и функции для реализации алгоритма поиска максимального из двух целых чисел.

```
program ProcMin;  
var  
    x, y, m: byte;  
procedure MinNum(a, b: byte; var min: byte);  
begin  
    if a < b then  
        min := a  
    else  
        min := b;  
end;
```

```
begin
  write('Введите x, y = ');
  readln(x, y);
  MinNum(x, y, m);
  writeln('min = ', m);
  readln;
end.

program FuncMin;
var
  x, y, m: byte;
function MinNum(a, b: byte): byte;
begin
  if a < b then
    MinNum := a
  else
    MinNum := b;
end;
begin
  write('Введите x, y = ');
  readln(x, y);
  m := MinNum(x, y);
  writeln('min = ', m);
  readln;
end.
```

5.14.3. Рекурсия

Рекурсия — это способ организации вычислительного процесса, при котором подпрограмма в ходе выполнения вызывает сама себя.

Рассмотрим программу **RecursiveFactorial**, в которой определена рекурсивная функция для вычисления факториала некоторого числа.

```
program RecursiveFactorial;
var
  n: longint;
function Fact(n: longint): longint;
begin
  if n = 0 then
    Fact := 1
  else
    Fact := n * Fact(n-1);
end;
```

```

begin
  write('n = ');
  readln(n);
  if n > 0 then
    writeln('n! = ', Fact(n))
  else
    writeln('Введите число больше 0');
  readln;
end.

```

Шаги выполнения программы **RecursiveFactorial**:

- 1) вводится число, факториал которого необходимо определить (предположим, что вычисляется факториал числа 2);
- 2) если значение переменной n меньше 0, то выводится сообщение:


```
writeln('Введите число больше 0');
```

 иначе вызывается функция `Fact`, в параметрах которой указывается введенное значение;
- 3) управление передается функции `Fact`: переменная n принимает значение 2;
- 4) выполняется анализ значения переменной n :
 - 4.1) если значение переменной n равно 1, то функция возвращает значение 1;
 - 4.2) если значение переменной n больше 1, то для расчета значения функции `Fact` используется выражение, в котором участвует значение переменной n и новый вызов функции `Fact`:


```
Fact := n * Fact(n-1);
```

На рис. 5.10 показан процесс рекурсивных вызовов функций `Fact` при вычислении факториала.

Шаг	Вызов функции
1	$F(2)$
2	$2 \cdot F(2 - 1)$
3	$1 \cdot F(1 - 1)$
4	$F(0) = 1$

Рекурсивный вызов может быть *косвенным*. В этом случае подпрограмма обращается к себе путем вызова другой подпрограммы, в которой уже содержится обращение к первой подпрограмме, например:

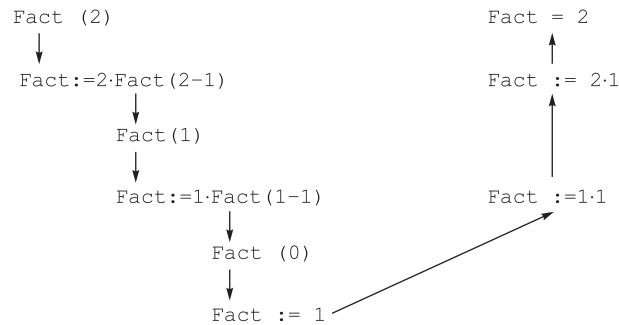


Рис. 5.10. Рекурсивные вызовы функции Fact

```

procedure B (j: byte); forward;
procedure A (i: byte);
begin
  <операторы>;
  B(i);
  <операторы>;
end;
procedure B;
begin
  <операторы>;
  A(j);
  <операторы>;
end;

```

В данной программе используется *опережающее описание*, которое заключается в том, что объявляется лишь заголовок процедуры *B*, а ее тело заменяется стандартной директивой **forward**. Это позволяет в процедуре *A* использовать обращение к процедуре *B*. Тело процедуры *B* при этом начинается заголовком, в котором уже не указываются описанные ранее формальные параметры.

5.14.4. Локальные и глобальные идентификаторы

Идентификаторы, описанные в заголовке или разделе описаний подпрограммы, называют *локальными для этой подпрограммы*. Идентификаторы же, описанные в основной программе, называют *глобальными*. Формальные параметры подпрограмм являются локальными переменными для соответствующих подпрограмм.

Особенности использования локальных и глобальных идентификаторов:

1) локальные идентификаторы доступны только внутри той подпрограммы, где они описаны;

2) идентификаторы, описанные в одной подпрограмме, могут совпадать с идентификаторами из других подпрограмм (как содержащих данную подпрограмму, так и вложенных в нее).

При входе в подпрограмму более низкого уровня не только становятся доступными объявленные в ней локальные идентификаторы, но и сохраняется доступ ко всем идентификаторам верхнего уровня («глобальным» для данной подпрограммы).

Пример использования локальных и глобальных идентификаторов:

```
program TestIdent;
var
  vrb1: byte;
procedure proc_1;
  var
    vrb2: byte;
  // начало proc_2
  procedure proc_2;
    var
      vrb3: byte;
    begin
      vrb3 := 3;
      vrb2 := 4;
      writeln('Вызов процедуры proc_2');
      writeln('vrb2 = ', vrb2);
      writeln('vrb3 = ', vrb3);
    end;
  // конец proc_2
  begin
    proc_2;
    writeln('Вызов процедуры proc_1');
    vrb2 := 2;
    writeln('vrb2 = ', vrb2);
  end;
begin
  proc_1;
  writeln('Вызов основной программы');
```

```
    vrb1 := 2;  
    writeln('vrb1 = ', vrb1);  
    readln;  
end.
```

Из процедуры `proc_2` доступны переменные *vrb1*, *vrb2* и *vrb3*, из процедуры `proc_1` доступны переменные *vrb1* и *vrb2*, а из основной программы — только переменная *vrb1*. При этом процедуру `proc_2` можно вызывать только из процедуры `proc_1`.

5.14.5. Передача в подпрограмму массивов и строк

Передача массива фиксированного размера или текстовой строки в подпрограмму выполняется следующим образом:

- 1) в разделе **type** объявляется тип массива или строки;
- 2) описанный тип используется в списке формальных и фактических параметров подпрограммы.

Программы **SendArray** и **SendString** демонстрируют передачу в подпрограмму массива и строки, соответственно.

```
program SendArray;  
const  
    n = 10;  
type  
    ArrType = array [1..n] of byte;  
var  
    i: byte;  
    ArrInProg: ArrType;  
procedure PrintArr(arr: ArrType);  
var  
    i: byte;  
begin  
    writeln('Массив:');  
    for i := 1 to n do  
        write(arr[i], ' ');  
end;  
begin  
    for i := 1 to n do  
        ArrInProg[i] := trunc(random(10));  
    PrintArr(ArrInProg);  
    readln;  
end.
```

```

program SendString;
const
    n = 10;
type
    StrType = string[n];
var
    i: byte;
    StrInProg: StrType;
procedure PrintStr(str: StrType);
begin
    writeln('Строка:');
    writeln(str);
end;
begin
    StrInProg := 'some str';
    PrintStr(StrInProg);
    readln;
end.

```

Способ передачи в подпрограмму массива произвольного размера основан на использовании *открытого параметра-массива*, который описывается следующим образом:

```
<имя_подпрограммы> ([var] OpenArr: array of <тип>);
```

В качестве открытых допускаются только одномерные массивы.

Границы переданного в подпрограмму массива можно определить, используя две функции — Low и High:

```

program OpenArray;
var
    arr: array [-2..3] of byte;
    ne: byte;
procedure TestOpen(inarr: array of byte);
var
    ne: byte;
begin
    writeln('Нижний индекс inarr: ', Low(inarr));
    writeln('Верхний индекс inarr: ', High(inarr));
    ne := High(inarr) - Low(inarr);
    writeln('Число элементов inarr: ', ne);
end;
begin
    writeln('Нижний индекс arr: ', Low(arr));
    writeln('Верхний индекс arr: ', High(arr));

```

```
    ne := High(arr) - Low(arr);  
    writeln('Число элементов arr: ', ne);  
    TestOpen(arr);  
    readln;  
end.
```

Результат выполнения этой программы:

```
Нижний индекс arr: -2  
Верхний индекс arr: 3  
Число элементов arr: 5  
Нижний индекс inarr: 0  
Верхний индекс inarr: 5  
Число элементов inarr: 5
```

Программа **FunAver** вычисляет среднее значение элементов массива чисел, используя открытый массив:

```
program FunAver;  
var  
    arr: array [-7..6] of single;  
    i: integer;  
    res: single;  
function Aver(var arr : array of single): single;  
var  
    i: integer;  
    s: single;  
begin  
    s := 0;  
    for i := Low(arr) to High(arr) do  
        s := s + arr[i];  
    Aver := s / (High(arr) - Low(arr));  
end;  
begin  
    for i := -7 to 6 do  
        arr[i] := random(100);  
    res := Aver(arr);  
    writeln('Среднее = ', res:5:3);  
    readln;  
end.
```

Для передачи в подпрограмму двумерного массива произвольного размера его можно преобразовать в одномерный, передать в подпрограмму, а в ней снова преобразовать в двумерный. Можно также использовать глобально объявленный двумерный массив.

Задачи для самостоятельного решения

Оператор присваивания, ввод-вывод данных

Задача 1. Определите значение переменной c после выполнения фрагмента программы:

```
a := 5;
b := 3;
a := a + b * 3;
if a > b then
    c := a - 2*b
else
    c := 2*b - a;
```

Задача 2. Определите значение переменной c после выполнения фрагмента программы:

```
a := 5;
b := 3;
a := a + b * 2;
if a < b then
    c := a - 2*b
else
    c := 2*b - a;
```

Задача 3. Определите значение целочисленных переменных x , y и t после выполнения фрагмента программы:

```
x := 3;
y := 5;
t := x * y;
x := y mod x;
y := t;
```

Задача 4. Определите значение целочисленных переменных x , y и t после выполнения фрагмента программы:

```
x := 3;
y := 5;
t := -1 * x * y;
x := y mod x;
y := -1*y;
```

Задача 5. Определите значение целочисленных переменных a и b после выполнения фрагмента программы:

```
a := 5851;  
b := (a mod 10)*1000;  
a := a div (10*b);
```

Задача 6. Вычислите значение выражения $\text{frac}(2.5) * (100 \text{ div } 10)$.

Задача 7. Вычислите значение выражения $\text{int}(2.7) + (11 \text{ mod } 3)$.

Задача 8. Определите значение целочисленных переменных a и b после выполнения фрагмента программы:

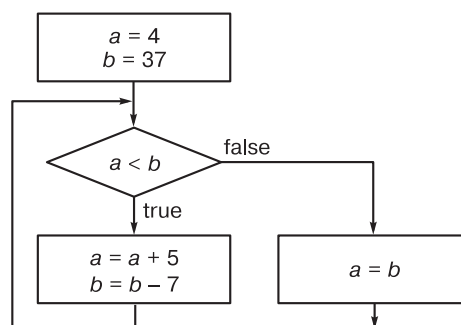
```
a := 333;  
b := 10 * a mod a;  
a := (b + 1) div 10;
```

Задача 9. Определите значение переменной f после выполнения фрагмента программы:

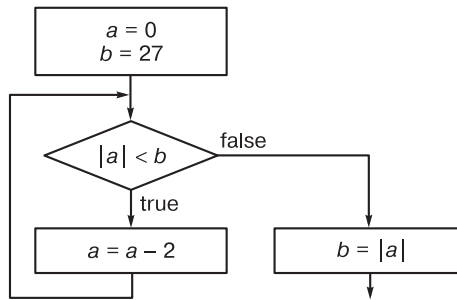
```
var  
  f: boolean;  
  x, y, z: char;  
begin  
  x := 'a';  
  y := 'c';  
  z := 'b';  
  f := true;  
  f := f xor ((x > z) or (y < x));  
end.
```

Структуры: циклическая и ветвление

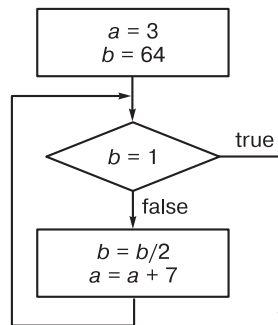
Задача 1. Запишите значение переменной a после выполнения фрагмента алгоритма:



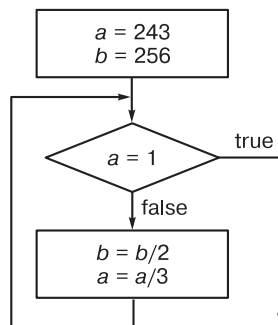
Задача 2. Запишите значение переменной a после выполнения фрагмента алгоритма:



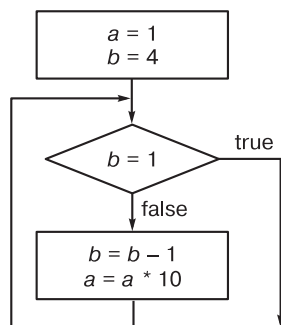
Задача 3. Запишите значение переменной a после выполнения фрагмента алгоритма:



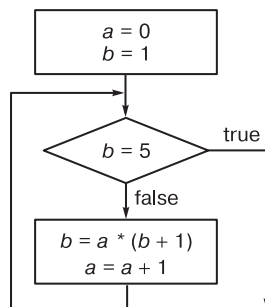
Задача 4. Запишите значение переменной b после выполнения фрагмента алгоритма:



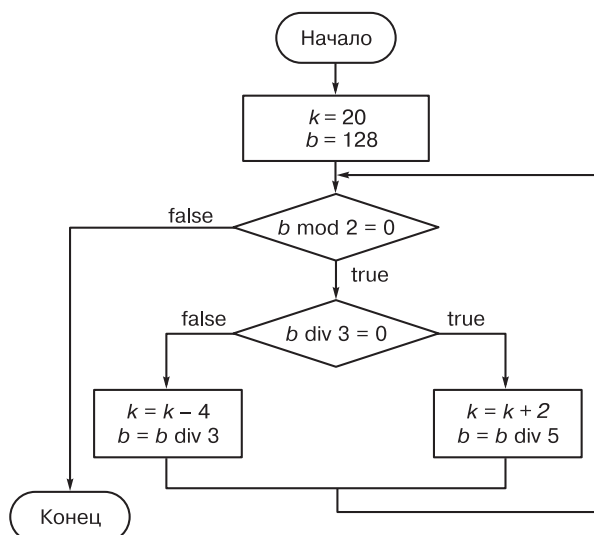
Задача 5. Запишите значение переменной a после выполнения фрагмента алгоритма:



Задача 6. Запишите значение переменной b после выполнения фрагмента алгоритма:



Задача 7. Запишите значение переменной k после выполнения фрагмента алгоритма:



Задача 8. В результате выполнения фрагмента программы:

```
while n <> 0 do
begin
  write (3 + (n mod 10));
  n := n div 10;
end;
```

было напечатано число 128126. Предполагается, что перед выполнением этого фрагмента алгоритма значение переменной n было равно 3959. Верно ли данное утверждение?

Задача 9. В результате выполнения фрагмента программы:

```
while n <> 0 do
begin
  write ((n mod 10) - 1);
  n := n div 10;
end;
```

было напечатано число 8473. Предполагается, что перед выполнением этого фрагмента алгоритма значение переменной n было равно 4859. Верно ли данное утверждение?

Задача 10. В результате выполнения фрагмента программы:

```
while n <> 0 do
begin
  write ((n mod 100)*2);
  n := n div 100;
end;
```

было напечатано число 16270. Предполагается, что перед выполнением этого фрагмента алгоритма значение переменной n было равно 3582. Верно ли данное утверждение?

Задача 11. Разработайте программу решения уравнения вида $a \cdot x^2 + b \cdot x + c = 0$. В программе предусмотрите все возможные сочетания значений параметров a , b и c .

Задача 12. Определите, сколько раз выполнится тело цикла в программе:

```
program SomeProgram;
var
  f: boolean;
  x: integer;
begin
  f := true;
```

```
x := 11;
while (f) do
begin
  f := ((x div 3) mod 2) > 0;
  x := x div 2;
end;
end.
```

Задача 13. Определите, чему равно значение переменной c , вычисляемое в программе:

```
program SomeProgram;
var
  m, c: integer;
begin
  c := 0;
  for m := 100 to 300 do
    if (m mod 15 = 0) and (m mod 35 <> 0)
      and (m mod 7 <> 0) then
      c := c + 1;
  writeln('c = ', c);
end.
```

Задача 14. Определите, чему равно значение переменной c , вычисляемое в программе:

```
program SomeProgram;
var
  m, c: integer;
begin
  c := 0;
  for m := 200 to 400 do
    if (m mod 20 = 0) and (m mod 30 <> 0)
      and (m mod 45 <> 0) then
      c := c + 1;
  writeln('c = ', c);
end.
```

Задача 15. Функция f вычисляется следующей программой:

```
function f(x:integer):integer;
begin
  if (x = 2) then
    f := 0
```

```
    else
      if (x mod 3 = 0) then
        f := 2 * f((x-1) div 2) + 2
      else
        f := 3 * f((x-1) div 2) + 1;
    end;
```

Чему равно значение функции $f(29)$?

Задача 16. Функция f вычисляется следующей программой:

```
function f(x:integer):integer;
begin
  if (x = 2) then
    f := 0
  else
    if (x mod 3 = 0) then
      f := 2 * f((x div 2) mod 3) + 4
    else
      f := 3 * f((x div 2) mod 3) + 2;
    end;
end;
```

Чему равно значение функции $f(28)$?

Работа с массивами

Задача 1. В программе используется одномерный целочисленный массив A с индексами от 0 до 10. Ниже представлен фрагмент программы, в котором значения элементов сначала задаются, а затем меняются:

```
for i := 0 to 10 do
  A[i] := 0;
for i := 0 to 10 do
  begin
    A[10-i] := A[10-i] + i;
  end;
```

Чему будут равны элементы этого массива после выполнения фрагмента программы?

Задача 2. В программе используется одномерный целочисленный массив A с индексами от 0 до 10. Ниже представлен фрагмент программы, в котором значения элементов сначала задаются, а затем меняются:

```
for i := 0 to 10 do
  begin
    A[i] := i;
  end;
for i := 0 to 10 do
  begin
    A[10-i] := A[i] + A[10-i];
  end;
```

Чему будут равны элементы этого массива после выполнения фрагмента программы?

Задача 3. Значения двумерного массива размера 8×8 задаются с помощью вложенного оператора цикла в представленном фрагменте программы:

```
for n := 1 to 8 do
  for k := 1 to 8 do
    B[n, k] := n - k;
```

Сколько элементов этого массива будут равны нулю?

Задача 4. Значения элементов двумерного массива размера 8×8 задаются в представленном фрагменте программы:

```
for n := 1 to 8 do
  for k := 1 to 8 do
    B[n, k] := power(-1, n*k) * n*k;
```

Сколько элементов этого массива будут больше нуля?

Задача 5. Значения элементов двумерного массива размера 8×8 задаются в представленном фрагменте программы:

```
for n := 1 to 8 do
  for k := 1 to 8 do
    B[n, k] := power(-1, n) + k + 1;
```

Сколько элементов этого массива будут больше нуля?

Задача 6. Значения двух массивов $A[1..10]$ и $B[1..10]$ задаются с помощью следующего фрагмента программы:

```
for n := 1 to 10 do
  A[n] := 5 - n;
for n := 1 to 10 do
  B[n] := A[n] * n;
```

Сколько элементов массива B будут иметь значение больше нуля?

Задача 7. Значения двух массивов $A[1..10]$ и $B[1..9]$ задаются с помощью следующего фрагмента программы:

```
for n:=1 to 10 do
  A[n] := n - 5;
for n:=1 to 9 do
  B[n] := A[n + 1];
```

Сколько элементов массива B будут иметь значение меньше нуля?

Задача 8. Значения двух массивов $A[1..10]$ и $B[1..10]$ задаются с помощью следующего фрагмента программы:

```
for n := 1 to 10 do
  A[n] := -1 * n * n + 3 * n;
for n := 1 to 10 do
  B[n] := -1 * A[n];
```

Сколько элементов массива B будут иметь значение больше нуля?

Задача 9. Определите, упорядочивает ли по возрастанию фрагмент программы:

```
for k := 1 to 3 do
  if (x[k] > x[7 - k]) then
    begin
      S := x[k];
      x[k] := x[7 - k];
      x[7 - k] := S;
    end;
```

массив (101, 32, 67, 59, 81, 23).

Задача 10. Дан фрагмент программы, обрабатывающей двумерный массив A размера $n \times n$:

```
k := 1;
l := 2;
for i:=1 to n do
  begin
    c := A[l, i];
    A[l, i] := A[k, i];
    A[k, i] := c;
  end;
```

Представьте массив в виде квадратной таблицы, в которой для элемента массива $A[i, j]$ величина i является номером строки, а величина j — номером столбца, в котором расположен элемент. Определите, что именно данный алгоритм меняет местами.

Задача 11. Дан целочисленный массив из 20 элементов. Каждый элемент из первой половины массива надо заменить его удвоенным значением, а каждый элемент из второй половины — уменьшить на единицу. Полученный массив вывести на экран.

Задача 12. Дан массив из 30 элементов. Надо рассчитать значение каждого элемента массива как сумму его собственного значения и значения предыдущего элемента, а первый элемент сложить с последним. Полученный массив вывести на экран.

Задача 13. Функцией `random(20)` заполнен массив чисел из 10 элементов. Написать программу для обмена значениями соседних элементов с четными и нечетными номерами.

Задача 14. Разработать программу, реализующую алгоритм нахождения наиболее часто встречающегося элемента в массиве $C(M, M)$.

Задача 15. Элементы целочисленного массива $a[1..5]$ равны (1, 2, 0, -1, 2). Определить значение выражения: $a[a[2]-a[4]*a[a[4]+3]]/a[2]$.

Задача 16. Значения элементов массива $p[1..3]$ равны $(a - 2*b, 3*b - 1, a \text{ div } b)$, где $a = 3, b = 1$. Определить значение выражения: $p[p[a]-p[a-b]+1]-p[p[a-2*b]+a-2*b]*2$.

Задание 17. Определить значение переменной m после выполнения приведенной ниже программы:

```
program SomeProgram;
var
  x: array[-1..1, -1..1] of integer;
  b: array[-1..1] of integer;
  i, j, m: integer;
begin
  for i := 1 to 3 do
  begin
    for j := 1 to 3 do
    begin
      x[i-2, j-2] := i*j mod 3;
    end;
    b[i-2] := i mod 3;
  end;
  m := 0;
  for i := -1 to 1 do
    for j := -1 to 1 do
```

```

        if x[i, b[j]-1] > x[b[i]-1, j] then
            m := m + 1;
        end.

```

Задание 18. Определить значение переменной m после выполнения приведенной ниже программы:

```

program SomeProgram;
var
x: array[-1..1, -1..1] of integer;
b: array[-1..1] of integer;
i, j, m: integer;
begin
    for i:=1 to 3 do
    begin
        for j:=1 to 3 do
        begin
            x[i-2, j-2] := i*j div 3;
        end;
        b[i-2] := (i + j - 1) mod 3 - 1;
    end;
    m := 1;
    for i:=-1 to 1 do
        for j:=-1 to 1 do
            if x[i, j] >= x[i, j] then
                m := m*(b[i]*b[j]) + 1;
        end;
    end.

```

Задача 19. Определить пропущенный член последовательности 2, 7, 12, ..., вычисляемой по приведенной ниже программе:

```

program SomeProgram;
var
    a, b, c: integer;
begin
    readln(a, b, c);
    if (b mod 2) = 0 then
        for i := 2 to 5 do
            begin
                write(c, ' ');
                c := c + a - b;
            end;
        end.

```


Задача 20. Определить пропущенный член последовательности 1, -5, -11, ..., вычисляемой по приведенной ниже программе:

```
program SomeProgram;
var
  a, b, c: integer;
begin
  readln(a, b, c);
  if (b mod 2) = 0 then
    for i := 2 to 5 do
      begin
        write(c, ' ');
        c := c + a*b;
      end;
  end.
```

Задача 21. Разработать программу, которая будет считывать с диска типизированный файл, содержащий вещественные числа, и выполнять их сортировку. Результат сортировки должен быть сохранен в текстовый файл.

Поиск ошибки в программе

Задача 1. Требовалось написать программу для определения, попадает ли точка $M(x, y)$ в круг радиусом R с центром в начале координат. Все числа считаются действительными. Программист торопился и написал программу неправильно. Приведите пример таких чисел x , y и R , при которых программа неверно решает поставленную задачу, и укажите, как нужно доработать программу, чтобы не было случаев ее неправильной работы.

```
program CircleTest;
var
  x, y, r: single;
begin
  {Ввод значения координаты x}
  write('Введите координату x: '); readln(x);
  {Ввод значения координаты y}
  write('Введите координату y: '); readln(y);
  {Ввод значения r - радиус окружности}
  write('Введите радиус r: '); readln(r);
  if (abs(x) <= r) or (abs(y) <= r) or
    (sqr(x) + sqr(y) <= sqr(r))
```

```
    then writeln('Точка принадлежит окружности')
else
    writeln('Точка вне окружности');
write('Нажмите Enter');
readln;
end.
```

Задача 2. Требовалось написать программу, которая по заданным координатам точки (x, y) в прямоугольной системе координат определяет, в какой четверти находится точка. Все числа считаются действительными. Программист торопился и написал программу неправильно. Приведите пример таких чисел x и y , при которых программа неверно решает поставленную задачу, и укажите, как нужно доработать программу, чтобы не было случаев ее неправильной работы.

```
program QuadrantTest;
var
    x, y: single;
begin
    writeln('Введите координаты точки:');
    write('x= ');
    readln(x);
    write('y= ');
    readln(y);
    if (x > 0) and (y < 0) then
        writeln('Точка находится в первой четверти');
    if (x < 0) and (y > 0) then
        writeln('Точка находится во второй четверти');
    if (x < 0) and (y < 0) then
        writeln('Точка находится в третьей четверти');
    if (x > 0) or (y < 0) then
        writeln('Точка находится в четвертой четверти');
    if (x = 0) or (y = 0) then
        writeln('Точка находится на оси координат');
end.
```

Задача 3. Требовалось написать программу, которая определяет количество слов в строке (строка заканчивается точкой; разделители слов — пробелы). Программист торопился и написал программу неправильно. Как нужно доработать программу, чтобы не было случаев ее неправильной работы?

```
program WordCount;
var
  s: string;
  k, i: integer;
begin
  write('введите строку:');
  readln(s);
  for i := 1 to length(s) do
    if (s[i] = ' ') and (s[i] = '.') then
      k := k + 2;
  writeln('количество слов в строке =', k);
end.
```

Задача 4. Требовалось написать программу, которая на интервале от 12 до 100 определяет числа, кратные 11, после чего вычисляет произведение этих чисел. Программист торопился и написал программу неправильно. Как нужно доработать программу, чтобы не было случаев ее неправильной работы?

```
program MulNumbers;
var
  i, j: integer;
  s: longint;
begin
  s := 1;
  writeln('Числа кратные 11 в интервале от 12 до 100:');
  for i := 12 to 100 do
    begin
      if i mod 11 = 0 then
        writeln(i);
        s := s * i;
    end;
  writeln('Произведение этих чисел: ', s);
  write('Нажмите Enter');
  readln;
end.
```

Задача 5. Требовалось написать программу для определения количества дней в месяце (ввод названия месяца производится с клавиатуры). Программист торопился и написал программу неправильно. Как нужно доработать программу, чтобы не было случаев ее неправильной работы?

```

program GetMonth;
var
    m: string;
    i, k: integer;
begin
    write('Введите интересующий Вас месяц: ');
    readln(m);
    if (m = 'Январь') or (m = 'Март') or (m = 'Май') or
        (m = 'Июль') or (m = 'Август') or (m = 'Октябрь')
        or m = 'Декабрь') then
        begin
            k := 31;
            if m = 'Февраль' then
                k := 28;
            if (m = 'Апрель') or (m = 'Июнь') or
                (m = 'Сентябрь') or (m = 'Ноябрь')
                then k := 30;
        end;
    writeln('В этом месяце ', k, ' дней');
    writeln('Нажмите Enter');
    readln;
end.

```

Задача 6. Требовалось написать функцию, которая возвращает минимальное из введенных чисел a, b, c, d . Программист торопился и написал функцию неправильно. Приведите пример таких чисел a, b, c, d , при которых функция неверно решает поставленную задачу, и укажите, как нужно доработать функцию, чтобы не было случаев ее неправильной работы.

```

program FindMin;
var
    a, b, c, d: single;
function min(a, b, c, d: single): single;
var
    tmp: single;
begin
    min := a;
    if (b > tmp) and (c > tmp) and (d > tmp) then tmp := b;
    if (a > tmp) and (c > tmp) and (d > tmp) then tmp := a;
    min := tmp;
end;

```

```
begin
  writeln('Введите числа a, b, c, d = ');
  readln(a, b, c, d);
  writeln('min(a, b, c, d) = ', min(a, b, c, d));
  readln;
end.
```

Задача 7. Требовалось написать программу определения корней квадратного уравнения $a \cdot x^2 + b \cdot x + c = 0$ ($a \neq 0, b \neq 0, c \neq 0$). Приведите пример таких чисел a, b, c , при которых программа неверно решает поставленную задачу, и укажите, как нужно доработать программу, чтобы не было случаев ее неправильной работы.

```
program SquareEquation;
var
  a, b, c, x1, x2, d: single;
begin
  write('Введите коэффициенты a, b, c: ');
  readln(a, b, c);
  d := (b * b) - (4 * a * c);
  if d <> 0 then
    writeln('Уравнение не имеет решений');
  if d = 0 then
    begin
      x1 := -b / (2 * a);
      writeln('x= ', x1:0:2);
    end;
  if d > 0 then
    begin
      x1 := (-b + sqrt(d)) / (2 * a);
    end;
  x2 := (-b - sqrt(d)) / (2 * a);
  writeln('x1 = ', x1:0:2, '; x2 = ', x2:0:2);
end.
```

Обработка текстовых данных

Задача 1. На вход программы подается текст на английском языке. Предложения могут заканчиваться символами '.', '!', '?' или ','. Требуется написать программу, которая будет в тексте определять и выводить на экран количество предложений каждого типа.

Задача 2. На вход программы подается файл, содержащий текст на английском языке. Предложения могут заканчиваться символа-

ми '.', '!' или '?'. Требуется написать программу, которая будет считывать текст из файла и определять, повторяется ли какое-либо предложение дважды, а если да, то выводить на экран это предложение.

Задача 3. На вход программы подается текст на английском языке. Предложения могут заканчиваться символами '.', '!' или '?'. Требуется написать программу, которая будет определять количество букв в самом длинном (по количеству букв) предложении.

Задача 4. На вход программы подается файл, содержащий текст на английском языке. Предложения могут заканчиваться символами '.', '!' или '?'. Требуется написать программу, которая будет выводить на экран предложения начиная с самого длинного (по количеству букв). Слова каждого предложения выводятся начиная с самого длинного по убыванию.

Задача 5. На вход программы подается текст на английском языке. Предложения могут заканчиваться символами '.', '!' или '?'. Требуется написать программу, которая будет выводить на экран все слова, встречающиеся в тексте, начиная с самого длинного.

Задача 6. На вход программе подаются сведения о заработной плате всех сотрудников некоторого учреждения за три месяца. В первой строке сообщается количество сотрудников N , а каждая из следующих N строк имеет формат:

```
<Фамилия> <Инициалы> <зарплата_1_мес> <зарплата_2_мес>  
<зарплата_3_мес>,
```

где <Фамилия> — строка, состоящая не более чем из 20 символов, <Инициалы> — строка, состоящая не более чем из 4 символов (буква, точка, буква, точка), <зарплата_1_мес>, <зарплата_2_мес> и <зарплата_3_мес> — числа. Записи <Фамилия> и <Инициалы>, а также <Инициалы> и каждое из чисел разделены одним пробелом. Пример входной строки:

```
Иванов П.С. 10000 12000 9100
```

Требуется написать программу, которая будет выводить на экран список сотрудников в порядке убывания их зарплаты в следующем виде:

```
Иванов П. С. 31100  
Петров К. В. 29200
```

Задача 7. На вход программы подаются произвольные алфавитно-цифровые символы. Ввод этих символов заканчивается точкой.

Требуется написать программу, которая будет выводить на экран входную строку с добавлением индекса к каждому символу, где индекс показывает, сколько раз данный символ до этого встречался в строке. Пример:

$ASDFAAHJ \Rightarrow A[0]S[0]D[0]F[0]A[1]A[2]H[0]J[0]$.

Задача 8. На вход программы подаются произвольные алфавитно-цифровые символы. Ввод этих символов заканчивается точкой. Требуется написать программу, которая будет выводить на экран входную строку с добавлением индекса к каждому символу, встречающемуся повторно, где индекс показывает, сколько раз данный символ до этого встречался в строке. Пример:

$ASDFAAHJ \Rightarrow ASDFA[1]A[2]HJ$.

Задача 9. Разработайте программу, выводящую на экран таблицу ASCII.

Задача 10. На вход программы подаются произвольные алфавитно-цифровые символы. Ввод этих символов заканчивается точкой. Требуется написать программу, которая будет проверять, является ли введенная последовательность симметричной. (Симметричной является последовательность, которая одинаково читается слева направо и справа налево, — *палиндром*.)

Задача 11. На вход программы подается строка. Определите сумму всех входящих в нее цифр.

Задача 12. На вход программы подается строка. Определите сумму всех входящих в нее целых чисел с учетом их знака.

Задача 13. На вход программы подается строка. Определите все символы, входящие в нее.

Задача 14. На вход программы подается строка. Определите наиболее часто встречающийся в ней символ.

Алгоритмы обработки данных

Задача 1. Разработайте программу, в которой реализуется линейный конгруэнтный генератор псевдослучайных чисел.

Задача 2. Разработайте программу, в которой реализуется линейный конгруэнтный генератор псевдослучайных чисел с одновременной модификацией последовательности генерируемых случайных чисел.

Задача 3. Разработайте программу, в которой реализовано два цикла, один из которых вложен в другой. Реализацию выполните с помощью оператора `goto`.

Задача 4. Разработайте программу перевода целого числа, находящегося в системе счисления с основанием a , в систему счисления с основанием b .

Подсказка: в качестве промежуточной можно использовать десятичную систему счисления.

Задача 5. Выполните программную реализацию алгоритма сортировки массива символов методом пузырька.

Задача 6. Выполните программную реализацию алгоритма сортировки массива символов методом быстрой сортировки.

Задача 7. Выполните программную реализацию алгоритма сортировки массива символов методом Шелла.

Задача 8. Выполните программную реализацию алгоритма сортировки массива символов методом Шейкера.

Задача 9. Выполните программную реализацию алгоритма сортировки массива символов методом вставки.

Задача 10. Выполните программную реализацию алгоритма сортировки массива символов методом слияния.

Задача 11. Выполните сортировку массива $A = [2, -2, 9, 2, 7, -5, 1]$ вручную методом пузырька. В ответе запишите последовательность выполняемых шагов.

Задача 12. Выполните сортировку массива $A = [9, -1, 9, 7, -4, 2, -7]$ вручную методом Шейкера. В ответе запишите последовательность выполняемых шагов.

Задача 13. Выполните сортировку массива $A = [-2, 6, -1, -10, 9, 0, 2]$ вручную методом Шелла. В ответе запишите последовательность выполняемых шагов.

Задача 14. Выполните сортировку массива $A = [-2, 6, -1, 10, 9, 0, -2]$ вручную методом слияния. В ответе запишите последовательность выполняемых шагов.

Задача 15. Выполните сортировку массива $A = [-4, -6, 0, 6, 7, -8, 4]$ вручную методом вставки. В ответе запишите последовательность выполняемых шагов.

Задача 16. Выполните сортировку массива $A = [-9, 6, 4, -4, -7, 0, 9]$ вручную методом Шейкера. В ответе запишите последовательность выполняемых шагов.

Задача 17. Разработайте программу, которая выполняет сортировку вещественных чисел, хранящихся в заданном файле, после чего записывает результат сортировки в другой файл.

Задача 18. На вход программы подается предложение, записанное в английском алфавите. Слова в предложении разделены пробелом, предложение заканчивается точкой. Разработайте программу для сортировки слов предложения методом Шейкера (при сортировке должен учитываться только первый символ слова).

Задача 19. Разработайте программу, выполняющую объединение двух отсортированных в порядке неубывания массивов. Результирующий массив также должен быть отсортирован по неубыванию элементов.

Задача 20. Разработайте программу, выполняющую сортировку дат, хранящихся в файле.

Задача 21. Пусть массив A содержит наименование марок автомобилей, а массив B — цены на соответствующие автомобили. Разработайте программу, выполняющую сортировку массива B по возрастанию с одновременной сортировкой массива A .

Задача 22. Разработайте программу для сжатия текстового файла с помощью метода Хаффмана. Выполните оценку степени сжатия.

Задача 23. Разработайте программу для сжатия звукового файла с помощью метода Хаффмана. Выполните оценку степени сжатия.

Задача 24. Разработайте программу для сжатия графического файла с помощью метода Хаффмана. Выполните оценку степени сжатия.

Задача 25. Выполните программную реализацию алгоритма кодирования сообщений на основе кодов Хэмминга (n, m) . Разработайте алгоритм внесения в исходную последовательность ошибок с заданными характеристиками.

Задача 26. Разработайте программу для кодирования/декодирования текстов на основе кодов Хэмминга (n, m) . Разработайте алгоритм внесения в изображение ошибок с заданными характеристиками.

Задача 27. Разработайте программу для кодирования/декодирования изображений с помощью кодов Хэмминга (n, m) . Разработайте алгоритм внесения в изображение ошибок с заданными характеристиками.

Задача 28. Разработайте программу для кодирования/декодирования звуковых записей с помощью кодов Хэмминга (n, m) . Разработайте алгоритм внесения в запись ошибок с заданными характеристиками.

Глава 6

Справочная информация

6.1. Действия со степенями

$$a^0 = 1; \quad a^{-n} = \frac{1}{a^n} \quad (a \neq 0);$$

$$a^m \cdot a^n = a^{m+n}; \quad a^m : a^n = a^{m-n} \quad (a \neq 0);$$

$$\sqrt[n]{a^m} = a^{\frac{m}{n}} \quad (n \neq 0, a > 0); \quad (a^m)^n = a^{m \cdot n}.$$

$$\text{Если } n, m \in \mathbb{N}, \text{ то } \sqrt[n]{a^{2m}} = \sqrt[n]{a^m}.$$

$$\text{Если } a > 0, b > 0, n \in \mathbb{N}, \text{ то } \sqrt[n]{a \cdot b} = \sqrt[n]{a} \cdot \sqrt[n]{b}; \quad \sqrt[n]{\frac{a}{b}} = \frac{\sqrt[n]{a}}{\sqrt[n]{b}}.$$

$$\text{Если } a \cdot b > 0, n \in \mathbb{N}, \text{ то } \sqrt[n]{a \cdot b} = \sqrt[n]{|a|} \cdot \sqrt[n]{|b|}; \quad \sqrt[n]{\frac{a}{b}} = \frac{\sqrt[n]{|a|}}{\sqrt[n]{|b|}}.$$

6.2. Логарифм числа и его свойства

Логарифм числа N по основанию a представляет собой показатель степени m , в которую следует возвести число a , чтобы получить N . Подобный логарифм обозначается как $\log_a N$, т. е. $m = \log_a N$, если $a^m = N$.

Основные свойства логарифма:

$$\log_a(M \cdot N) = \log_a M + \log_a N; \quad \log_a \frac{M}{N} = \log_a M - \log_a N.$$

$$\text{Например, } \log_3 81 = 3, \quad \log_2 32 = 5.$$

Логарифм числа 16 по основанию 2 ($\log_2 16$) есть число 4: возводя основание логарифма 2 в степень 4, мы получим число 16, логарифм которого вычисляется.

6.3. Таблицы сложения и умножения

В двоичной системе счисления

+	0	1
0	0	1
1	1	10

×	0	1
0	0	0
1	0	1

В восьмеричной системе счисления

+	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	10
2	2	3	4	5	6	7	10	11
3	3	4	5	6	7	10	11	12
4	4	5	6	7	10	11	12	13
5	5	6	7	10	11	12	13	14
6	6	7	10	11	12	13	14	15
7	7	10	11	12	13	14	15	16

×	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	10	12	14	16
3	0	3	6	11	14	17	22	25
4	0	4	10	14	20	24	30	34
5	0	5	12	17	24	31	36	43
6	0	6	14	22	30	36	44	52
7	0	7	16	25	34	43	52	61

В шестнадцатеричной системе счисления

+	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10
2	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11
3	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12
4	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13
5	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14
6	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15
7	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16
8	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17
9	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18
A	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19
B	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A
C	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B
D	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C
E	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D
F	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E

×	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2	0	2	4	6	8	A	C	E	10	12	14	16	18	1A	1C	1E
3	0	3	6	9	C	F	12	15	18	1B	1E	21	24	27	2A	2D
4	0	4	8	C	10	14	18	1C	20	24	28	2C	30	34	38	3C
5	0	5	A	F	14	19	1E	23	28	2D	32	37	3C	41	46	4B
6	0	6	C	12	18	1E	24	2A	30	36	3C	42	48	4E	54	5A
7	0	7	E	15	1C	23	2A	31	38	3F	46	4D	54	5B	62	69
8	0	8	10	18	20	28	30	38	40	48	50	58	60	68	70	78
9	0	9	12	1B	24	2D	36	3F	48	51	5A	63	6C	75	7E	87
A	0	A	14	1E	28	32	3C	46	50	5A	64	6E	78	82	8C	96
B	0	B	16	21	2C	37	42	4D	58	63	6E	79	84	8F	9A	A5
C	0	C	18	24	30	3C	48	54	60	6C	78	84	90	9C	A8	B4
D	0	D	1A	27	34	41	4E	5B	68	75	82	8F	9C	A9	B6	C3
E	0	E	1C	2A	38	46	54	62	70	7E	8C	9A	A8	B6	C4	D2
F	0	F	1E	2D	3C	4B	5A	69	78	87	96	A5	B4	C3	D2	E1

6.4. Функции электронных таблиц

Функции для работы с датой и временем

Excel	Calc	Описание
ДАТА	DATE	Возвращает числовую форму представления заданной даты
ДАТАЗНАЧ	DATEVALUE	Преобразует дату из текстовой формы в числовую
ДЕНЬ	DAY	Возвращает номер дня в месяце для даты, заданной в числовой форме
ДНЕЙ360	DAYS360	Вычисляет количество дней между двумя датами на основе 360-дневного года
ДАТАМЕС	EDATE	Возвращает числовую форму записи даты, отстоящей на заданное число месяцев вперед или назад от начальной даты
КОНМЕСЯЦА	EOMONTH	Возвращает числовую форму для даты последнего дня месяца, отстоящего вперед или назад на заданное число месяцев

Excel	Calc	Описание
ЧАС	HOUR	Возвращает час, соответствующий заданному времени в числовом формате
МИНУТЫ	MINUTE	Возвращает минуту, соответствующую заданному времени в числовом формате
МЕСЯЦ	MONTH	Возвращает номер месяца, соответствующий заданному времени в числовом формате
ЧИСТРАВДНИ	NETWORKDAYS	Возвращает числовую формулу всем рабочим дням между двумя датами
ТДАТА	NOW	Возвращает числовую формулу для текущих даты и времени
СЕКУНДЫ	SECOND	Возвращает секунду, соответствующую заданному времени в числовом формате
ВРЕМЯ	TIME	Возвращает числовую формулу представления заданного времени
ВРЕМЗНАЧ	TIMEVALUE	Преобразует время из текстового формата в числовую формулу
СЕГОДНЯ	TODAY	Возвращает числовую формулу для текущей даты
ДЕНЬНЕД	WEEKDAY	Возвращает номер дня недели для даты, заданной в числовом формате
НОМНЕДЕЛИ	WEEKNUM	Преобразует числовую формулу в число, указывающее, на какую неделю года приходится указанная дата
РАВДЕНЬ	WORKDAY	Возвращает числовую формулу по дате, отстоящей вперед или назад на заданное количество рабочих дней
ГОД	YEAR	Возвращает номер года для даты, заданной в числовом формате
ДОЛЯГОДА	DAYS	Возвращает долю года, которую составляет количество дней между начальной и конечной датами

Инженерные функции (функции для анализа)

Excel	Calc	Описание
ДВ.В.ДЕС	BIN2DEC	Преобразует двоичное число в десятичное
ДВ.В.ШЕСТН	BIN2HEX	Преобразует двоичное число в шестнадцатеричное
ДВ.В.ВОСЬМ	BIN2OCT	Преобразует двоичное число в восьмеричное
ПРЕОБР	—	Преобразует число из одной системы счисления в другую
ДЕС.В.ДВ	DEC2BIN	Преобразует десятичное число в двоичное
ДЕС.В.ШЕСТН	DEC2HEX	Преобразует десятичное число в шестнадцатеричное
ДЕС.В.ВОСЬМ	DEC2OCT	Преобразует десятичное число в восьмеричное
ПОРОГ	GESTEP	Проверяет, не превышает ли число пороговое значение
ШЕСТН.В.ДВ	HEX2BIN	Преобразует шестнадцатеричное число в двоичное
ШЕСТН.В.ДЕС	HEX2DEC	Преобразует шестнадцатеричное число в десятичное
ШЕСТН.В.ВОСЬМ	HEX2OCT	Преобразует шестнадцатеричное число в восьмеричное
ВОСЬМ.В.ДВ	OCT2BIN	Преобразует восьмеричное число в двоичное
ВОСЬМ.В.ДЕС	OCT2DEC	Преобразует восьмеричное число в десятичное
ВОСЬМ.В.ШЕСТН	OCT2HEX	Преобразует восьмеричное число в шестнадцатеричное

Функции проверки свойств и значений

Excel	Calc	Описание
ЯЧЕЙКА	CELL	Возвращает сведения о форматировании, местоположении или о содержимом заданной ячейки
ТИП.ОШИБКИ	ERRORTYPE	Возвращает номер, соответствующий типу ошибки
ИНФОРМ	INFO	Возвращает сведения о текущей операционной среде
ЕПУСТО	ISBLANK	Возвращает значение «ИСТИНА» для пустого значения
ЕОШ	ISERR	Возвращает значение «ИСТИНА» для любого ошибочного значения, кроме «#Н/Д»
ЕОШИБКА	ISERROR	Возвращает значение «ИСТИНА» для любого ошибочного значения
ЕЧЕТН	ISEVEN_ADD	Возвращает значение «ИСТИНА» для четного числа
ЕЛОГИЧ	ISLOGICAL	Возвращает значение «ИСТИНА» для логического значения
ЕНД	ISNA	Возвращает значение «ИСТИНА» для ошибочного значения «#Н/Д»
ЕНЕТЕКСТ	ISNONTEXT	Возвращает значение «ИСТИНА» для нетекстового значения (либо пустой ячейки)
ЕЧИСЛО	ISNUMBER	Возвращает значение «ИСТИНА» для числового значения
ЕССЫЛКА	ISREF	Возвращает значение «ИСТИНА» для значения, являющегося ссылкой
ЕТЕКСТ	ISTEXT	Возвращает значение «ИСТИНА» для текстового значения
Ч	N	Возвращает значение, преобразованное в число
НД	NA	Возвращает значение ошибки «#Н/Д»
ТИП	TYPE	Возвращает число, указывающее тип данных или значения

Логические функции

Excel	Calc	Описание
И	AND	Возвращает значение «ИСТИНА», если все аргументы имеют значение «ИСТИНА»
ЛОЖЬ	FALSE	Возвращает логическое значение «ЛОЖЬ»
ЕСЛИ	IF	Проверяет логическое значение для заданного условия и возвращает одно из двух заданных значений
НЕ	NOT	Меняет логическое значение аргумента на противоположное («ИСТИНА» на «ЛОЖЬ» и наоборот)
ИЛИ	OR	Возвращает значение «ИСТИНА», если хотя бы один аргумент имеет значение «ИСТИНА»
ИСТИНА	TRUE	Возвращает логическое значение «ИСТИНА»

Функции ссылки и поиска («Ссылки и массивы»)

Excel	Calc	Описание
АДРЕС	ADDRESS	Возвращает адрес ячейки в виде текста по заданным номерам строки и столбца
ОБЛАСТИ	AREAS	Возвращает количество областей в ссылке (область — это обособленные отдельная ячейка или интервал смежных ячеек)
СТОЛБЕЦ	COLUMN	Возвращает номер столбца для заданной ссылки на ячейку (без аргумента — возвращает номер текущего столбца)
ЧИСЛСТОЛБ	COLUMNS	Возвращает количество столбцов в ссылке или массиве
ГПР	HLOOKUP	Ищет в первой строке массива заданное значение, а затем ищет и возвращает значение из ячейки, находящейся в найденном столбце и заданной строке массива

Excel	Calc	Описание
ГИПЕРССЫЛКА	HYPERLINK	Создает ярлык или переход, открывающий документ, хранящийся на сетевом сервере, во внутренней сети или в Интернете
ИНДЕКС	INDEX	Для заданной ссылки или массива ищет и возвращает значение, находящееся на пересечении заданных порядковыми номерами строки и столбца относительно верхнего угла заданного диапазона (есть две формы записи этой функции для работы со ссылками и массивами)
СМЕЩ	OFFSET	Возвращает ссылку на ячейку или диапазон, отстоящие на заданное количество строк и столбцов от заданной ссылкой ячейки или диапазона
СТРОКА	ROW	Возвращает номер строки для заданной ссылки на ячейку (без аргумента — возвращает номер текущей строки)
ЧСТРОК	ROWS	Возвращает количество строк в ссылке или массиве
ТРАНСП	TRANSPOSE	Транспонирует массив (превращает строки в столбцы и наоборот; аналогично повороту массива на 90°)
ВПР	VLOOKUP	Ищет в первом столбце массива заданное значение, а затем ищет и возвращает значение из ячейки, находящейся в найденной строке и заданном столбце массива

Математические и тригонометрические функции

Excel	Calc	Описание
ABS	ABS	Возвращает абсолютное значение числа (модуль)
ACOS	ACOS	Возвращает арккосинус числа (в радианах)
ASIN	ASIN	Возвращает арксинус числа (в радианах)

Excel	Calc	Описание
ATAN	ATAN	Возвращает арктангенс числа (в радианах)
ATAN2	ATAN2	Возвращает арктангенс для заданных координат x и y (в радианах)
ОКРВВЕРХ	CEILING	Округляет число до ближайшего целого или до ближайшего кратного (округление с избытком)
ЧИСЛКОМБ	COMBIN	Возвращает количество комбинаций для заданного числа объектов
COS	COS	Возвращает косинус заданного угла в радианах
ГРАДУСЫ	DEGREES	Преобразует радианы в градусы
ЧЁТН	EVEN	Округляет число до ближайшего четного целого (положительные — в сторону увеличения, отрицательные — в сторону уменьшения)
EXP	EXP	Возвращает число e , возведенное в указанную степень (экспонента числа)
ФАКТР	FACT	Возвращает факториал числа
НОД	GCD	Возвращает наибольший общий делитель
ЦЕЛОЕ	INT	Округляет число до ближайшего меньшего целого
НОК	LCM	Возвращает наименьшее общее кратное
LN	LN	Возвращает натуральный логарифм числа
LOG	LOG	Возвращает логарифм числа по заданному основанию
LOG10	LOG10	Возвращает десятичный логарифм числа
ОСТАТ	MOD	Возвращает остаток от деления
ОКРУГЛТ	CEILING	Возвращает число, округленное до ближайшего кратного
НЕЧЁТ	ODD	Округляет число до ближайшего нечетного целого (положительные — в сторону увеличения, отрицательные — в сторону уменьшения)

Excel	Calc	Описание
ПИ	PI	Возвращает значение «пи» (с точностью 15 знаков после запятой)
СТЕПЕНЬ	POWER	Возвращает результат возведения числа в степень
ПРОИЗВЕД	PRODUCT	Перемножает аргументы
ЧАСТНОЕ	QUOTIENT	Возвращает целую часть результата деления
РАДИАНЫ	RADIANS	Преобразует градусы в радианы
СЛЧИС	RAND	Возвращает случайное число от 0 до 1 (равномерное распределение)
СЛУЧМЕЖДУ	RANDBETWEEN	Возвращает случайное число из интервала между заданными числами
ОКРУГЛ	ROUND	Округляет число до указанного числа десятичных разрядов
ОКРУГЛВНИЗ	ROUNDDOWN	Округляет число до ближайшего меньшего по модулю
ОКРУГЛВВЕРХ	ROUNDUP	Округляет число до ближайшего большего по модулю
ЗНАК	SIGN	Возвращает знак числа
SIN	SIN	Возвращает синус заданного угла в радианах
КОРЕНЬ	SQRT	Возвращает положительное значение квадратного корня
КОРЕНЬПИ	SQRTPI	Возвращает квадратный корень произведения заданного числа на число π
СУММ	SUM	Суммирует аргументы
СУММЕСЛИ	SUMIF	Суммирует ячейки, удовлетворяющие заданному условию
СУММКВ	SUMSQ	Вычисляет сумму квадратов аргументов
TAN	TAN	Возвращает тангенс угла в радианах
ОТБР	TRUNC	Отбрасывает дробную часть числа

Статистические функции

Excel	Calc	Описание
СРОТКЛ	AVEDEV	Возвращает среднее арифметическое абсолютных отклонений точек данных от среднего
СРЗНАЧ	AVERAGE	Возвращает среднее арифметическое аргументов
СРЗНАЧА	AVERAGEA	Возвращает среднее арифметическое аргументов, включая числа, текст и логические значения
СЧЁТ	COUNT	Подсчитывает количество чисел в списке аргументов
СЧЁТЗ	COUNTA	Подсчитывает количество непустых значений в списке аргументов
СЧЁТЕСЛИ	COUNTIF	Подсчитывает количество ячеек в диапазоне, удовлетворяющих заданному условию
МАКС	MAX	Возвращает максимальное значение из списка числовых аргументов (логические значения и текст игнорируются)
МАКСА	MAXA	Возвращает максимальное значение из списка аргументов, включая числа, текст и логические значения
МЕДИАНА	MEDIAN	Возвращает медиану заданных чисел
МИН	MIN	Возвращает минимальное значение из списка числовых аргументов (логические значения и текст игнорируются)
МИНА	MINA	Возвращает минимальное значение из списка аргументов, включая числа, текст и логические значения
ПЕРЕСТ	PERMUT	Возвращает количество перестановок для заданного числа объектов

Функции для работы с текстом

Excel	Calc	Описание
СИМВОЛ	CHAR	Возвращает символ по его коду (для ОС Windows — кодировка ANSI)
КОДСИМВ	CODE	Возвращает числовой код первого символа в текстовой строке (для ОС Windows — кодировка ANSI)

Excel	Calc	Описание
СЦЕПИТЬ	CONCATENATE	Объединяет несколько текстовых строк в одну
СОВПАД	EXACT	Сравнивает две текстовые строки и возвращает значение «ИСТИНА», если они полностью совпадают
ЛЕВСИМВ	LEFT	Возвращает указанное количество символов с начала текстовой строки (слева)
ДЛСТР	LEN	Возвращает количество знаков в текстовой строке
СТРОЧН	LOWER	Преобразует буквы текста в строчные
ПРОПНАЧ	PROPER	Делает прописной первую букву каждого слова в тексте
ЗАМЕНИТЬ	REPLACE	Заменяет заданную часть текста на другой текстовый фрагмент
ПОВТОР	REPT	Повторяет текст заданное число раз
ПРАВСИМВ	RIGHT	Возвращает указанное количество символов с конца текстовой строки (справа)
ПОИСК	FIND	Ищет одно текстовое значение внутри другого (без учета регистра) и возвращает номер позиции первого вхождения исходной строки или символа в исходной строке. При поиске можно использовать <i>знаки шаблона</i> — * и ?
ПОДСТАВИТЬ	SUBSTITUTE	Заменяет в текстовой строке старый текст новым
Т	T	Проверяет, является ли аргумент текстом или ссылкой на текст. Если да, то возвращает этот текст, иначе возвращает пустую строку
ТЕКСТ	TEXT	Форматирует число и преобразует его в текст
СЖПРОБЕЛЫ	TRIM	Удаляет из текста лишние пробелы (оставляя по одному пробелу между словами)
ПРОПИСН	UPPER	Преобразует буквы текста в прописные
ЗНАЧЕН	VALUE	Преобразует текстовую строку в число (отсекая нечисловые символы)

6.5. Коды ASCII

Управляющие символы*

Код	Обозначение	Клавиша	Значение	Отображаемый символ
0	nul	^@	Нуль	
1	soh	^A	Начало заголовка	☺
2	stx	^B	Начало текста	☹
3	etx	^C	Конец текста	♥
4	eot	^D	Конец передачи	♦
5	enq	^E	Запрос	♣
6	ack	^F	Подтверждение	♠
7	bel	^G	Сигнал («звонок») встроенного пьезодинамика	•
8	bs	^H	Забой (шаг назад)	▣
9	ht	^I	Горизонтальная табуляция	○
10	lf	^J	Перевод строки	▣
11	vt	^K	Вертикальная табуляция	♂
12	ff	^L	Новая страница	♀
13	cr	^M	Возврат каретки	♪
14	so	^N	Выключить сдвиг	♪
15	si	^O	Включить сдвиг	☼
16	dle	^P	Ключ связи данных	▶
17	dc1	^Q	Управление устройством 1	◀
18	dc2	^R	Управление устройством 2	↕
19	dc3	^S	Управление устройством 3	!!
20	dc4	^T	Управление устройством 4	¶
21	nak	^U	Отрицательное подтверждение	§
22	syn	^V	Синхронизация	—
23	etb	^W	Конец передаваемого блока	↕
24	can	^X	Отказ	↑
25	em	^Y	Конец среды	↓
26	sub	^Z	Замена	→

* Визуальное представление этих кодов в виде того или иного символа никак не связано с его значением: просто в разных ситуациях данный код либо вызывает соответствующее действие («значение»), либо отображается в виде соответствующего символа. — *Прим. ред.*

Код	Обозначение	Клавиша	Значение	Отображаемый символ
27	esc	^[Ключ	←
28	fs	^\	Разделитель файлов	└
29	gs	^]	Разделитель группы	↔
30	rs	^^	Разделитель записей	▲
31	us	^_	Разделитель модулей	▼

Символы с кодами 32–127

Код	Символ	Код	Символ	Код	Символ	Код	Символ
32	пробел	56	8	80	P	104	h
33	!	57	9	81	Q	105	i
34	"	58	:	82	R	106	j
35	#	59	;	83	S	107	k
36	\$	60	<	84	T	108	l
37	%	61	=	85	U	109	m
38	&	62	>	86	V	110	n
39	`	63	?	87	W	111	o
40	(64	@	88	X	112	p
41)	65	A	89	Y	113	q
42	*	66	B	90	Z	114	r
43	+	67	C	91	[115	s
44	,	68	D	92	\	116	t
45	-	69	E	93]	117	u
46	.	70	F	94	^	118	v
47	/	71	G	95	_	119	w
48	0	72	H	96	`	120	x
49	1	73	I	97	a	121	y
50	2	74	J	98	b	122	z
51	3	75	K	99	c	123	{
52	4	76	L	100	d	124	
53	5	77	M	101	e	125	}
54	6	78	N	102	f	126	~
55	7	79	O	103	g	127	DEL

Символы с кодами 128–255 (кодовая таблица CP-866)

Код	Символ	Код	Символ	Код	Символ	Код	Символ
128	А	160	а	192	Ќ	224	р
129	Б	161	б	193	Ћ	225	с
130	В	162	в	194	Ў	226	т
131	Г	163	г	195	ѐ	227	у
132	Д	164	д	196	—	228	ф
133	Е	165	е	197	†	229	х
134	Ж	166	ж	198	‡	230	ц
135	З	167	з	199	‖	231	ч
136	И	168	и	200	ℓ	232	ш
137	Й	169	й	201	ѓ	233	щ
138	К	170	к	202	Ⓙ	234	ъ
139	Л	171	л	203	џ	235	ы
140	М	172	м	204	ѣ	236	ь
141	Н	173	н	205	=	237	э
142	О	174	о	206	†	238	ю
143	П	175	п	207	±	239	я
144	Р	176	␣	208	ℓ	240	Е
145	С	177	␣	209	џ	241	е
146	Т	178	␣	210	π	242	Є
147	У	179		211	ℓ	243	є
148	Ф	180	†	212	ℓ	244	İ
149	Х	181	‡	213	ѓ	245	ı
150	Ц	182	‖	214	π	246	Ÿ
151	Ч	183	π	215	†	247	ÿ
152	Ш	184	‡	216	†	248	°
153	Щ	185	‖	217	Ј	249	•
154	Ъ	186	‖	218	г	250	•
155	Ы	187	ѓ	219	■	251	√
156	Ь	188	Ј	220	■	252	№
157	Э	189	Ј	221	■	253	¤
158	Ю	190	Ј	222	■	254	■
159	Я	191	г	223	■	255	

Символы с кодами 128–255 (кодовая таблица CP-1251)

Код	Символ	Код	Символ	Код	Символ	Код	Символ
128	Ѣ	160		192	А	224	а
129	Ѓ	161	Ў	193	Б	225	б
130	҃	162	Ѳ	194	В	226	в
131	҃	163	Ј	195	Г	227	г
132	„	164	Ѣ	196	Д	228	д
133	...	165	Ѓ	197	Е	229	е
134	†	166	Ї	198	Ж	230	ж
135	‡	167	§	199	З	231	з
136	€	168	Е	200	И	232	и
137	‰	169	©	201	Й	233	й
138	Љ	170	Є	202	К	234	к
139	<	171	«	203	Л	235	л
140	Њ	172	¬	204	М	236	м
141	Ќ	173	–	205	Н	237	н
142	Њ	174	®	206	О	238	о
143	Џ	175	İ	207	П	239	п
144	ђ	176	°	208	Р	240	р
145	`	177	±	209	С	241	с
146	'	178	І	210	Т	242	т
147	“	179	ı	211	У	243	у
148	”	180	Г	212	Ф	244	ф
149	•	181	μ	213	Х	245	х
150	—	182	¶	214	Ц	246	ц
151	—	183	•	215	Ч	247	ч
152	—	184	е	216	Ш	248	ш
153	™	185	№	217	Щ	249	щ
154	Љ	186	е	218	Ъ	250	ъ
155	>	187	»	219	Ы	251	ы
156	Њ	188	ј	220	Ь	252	ь
157	Ќ	189	Ѕ	221	Э	253	э
158	ћ	190	Ѕ	222	Ю	254	ю
159	џ	191	ї	223	Я	255	я

Ответы

Глава 1

Методы измерения количества информации

1. 2^{26} . 2. 3. 3. 243. 4. 4. 5. 500. 6. 12. 7. 62. 8. 12. 9. 14. 10. 9.

Процесс передачи информации

1. 254 Кбит/с. 2. 702 Кбит/с. 3. 1260 Кб. 4. 12 с. 5. 5136 с.

Представление числовой информации

1. 2. 2. 9. 3. 6. 4. 7. 5. 3. 6. 4. 7. Выполняется. 8. Выполняется.
9. 8.5. 10. 420.5. 11. 522.5. 12. $1.001(011)_2$. 13. $3.26(52)_8$. 14. $2.7(1)_8$.
15. $1.77(5673)_8$. 16. 10. 17. 400 байт. 18. 160 байт. 19. 7A20. 20. 3637.
21. 2D. 22. 4762. 23. 24F8. 24. bccda. 25. qbebl. 26. Верно. 27. Не-
верно. 28. 6. 29. 8. 30. 8. 31. 2. 32. 5. 33. 2. 34. 11. 35. 12, 37, 62,
87. 36. 8. 37. 4. 38. 4, 7, 14, 28. 39. 5, 10. 40. 9. 41. 5. 42. (0,2,5).
43. (2,1,4). 44. $x = 5$. 45. $x = 16$. 46. $x = 17$. 47. 10 Вариантов. 48. 8 ва-
риантов. 49. 12BC. 50. AF11. 51. 26431. 52. 0110011. 53. 1110000.
54. 1111111. 55. 1. 56. 7. 58. $a = 0$, пробел = 111, $b = 110$, $c = 101$,
 $d = 1001$, $f = 1000$, до — 60 бит, после — 46 бит. 59. точка = 10,
2 = 00, 3 = 110, 0 = 011, 1 = 010, 4 = 1110, 5 = 11110, 7 = 111111,
8 = 111110, до — 104 бит, после — 77 бит. 60. $m = 10$, точка = 01,
- = 00, $a = 1111$, $s = 1110$, $d = 1101$, + = 1010, 7 = 111111, 8 = 111110,
до — 66 бит, после — 58 бит.

Представление чисел в прямом, обратном и дополнительных кодах

1. -113. 2. -63. 3. 10011100. 4. 10011011. 5. Дополнительный
код: 11110010, обратный код: 11110001. 6. 4. 7. 3. 8. $10010 + 111 =$
 $= 11001$. 9. $F1C1 + 123 = F2E4$.

Представление чисел с фиксированной и плавающей запятой

1. 70352. 2. 71C7. 3. 121. 4. -206. 5. 106. 6. 55. 7. 505. 8. 43E74000.
9. 411C0000. 10. 43678000.

Арифметические операции над числами

1. 1000111_2 . 2. 332_8 . 3. 1111110_2 . 4. 3203. 5. 2. 6. 14. 7. 0. 8. 11111.
9. 26. 10. 4. 11. 11110. 12. 132.15. 13. 30.9. 14. 5.125.

Кодирование текстовой информации

1. 48 байт. 2. 22. 3. 73. 4. 20 байт. 5. 40. 6. 70. 7. 65536. 8. 2 Кб. 9. 7А.
10. 700. 11. 1. 12. home. 13. SPORT. 14. 200 237 244 238 240 236 224
242 232 234 224.

Кодирование графической информации

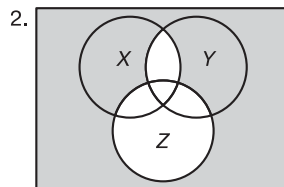
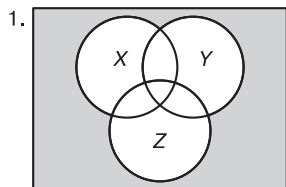
1. 4. 2. 2,25 Мб. 3. 2 Кб. 4. 90. 5. 48. 6. Желтый. 7. Серый. 8. 2. 9. 4.
10. 8. 11. 12.

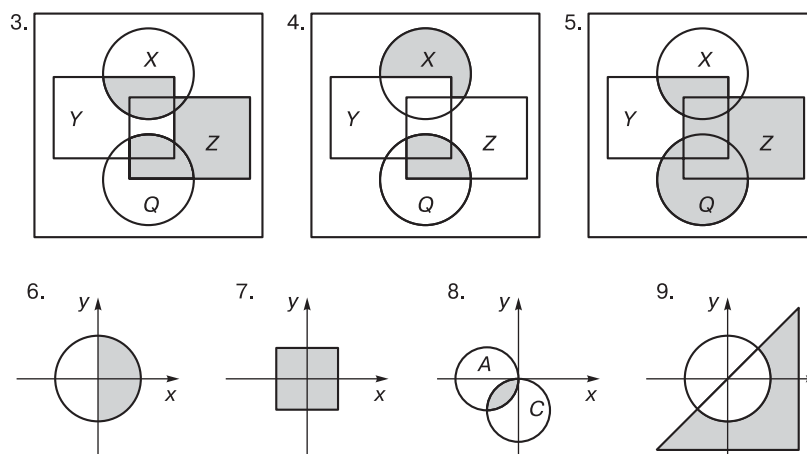
Кодирование звуковой информации

1. 125 Кб. 2. 31.25 Кб. 3. 6 с. 4. 16 с. 5. 8 кГц. 6. 16 кГц.
7. 2.5 с. 8. 0.625 с.

Комбинаторика

1. $81 + 81 + 81 + 72 + 72 + 72 = 459$. 2. $2^{I_n} = 120$. 3. $500 + 500 + 500 + 625 = 2125$. 4. $500 + 500 + 626 + 626 + 500 + 626 + 500 + 500 + 500 + 626 = 6000$. 5. $20 + 320 + 400 + 400 + 320 + 400 = 2160$. 6. $9 \cdot 10 \cdot 10 \cdot 5 = 4500$. 7. $125 + 75 + 60 + 48 = 308$.
8. $4 \cdot 9 + 36 + 36 = 108$. 9. $504 + 448 + 448 + 448 = 1848$. 10. $9 \cdot 10 \cdot 10 \times 2 = 1800$. 11. $P_2 \cdot P_6 = 1440$. 12. $A_5^3 + A_5^4 + A_5^5 = 300$. 13. $3 \cdot P_6 = 2160$.
14. $C_{20}^3 = 1140$. 15. $A_6^4 = 360$. 16. $A_{11}^5 = 55440$. 17. $P_7 - P_2 \cdot P_6 = 3600$.
18. $P_8 = 40320$.

Теория множеств



Глава 2

Логические выражения и их преобразование

1. Да. 2. Нет. 3. $P = \text{TRUE}$, $Q = \text{TRUE}$. 4. $P = \text{FALSE}$, $Q = \text{FALSE}$. 5. 1) $P = \text{FALSE}$, $Q = \text{FALSE}$; 2) $P = \text{TRUE}$, $Q = \text{TRUE}$. 6. $P = \text{FALSE}$, $Q = \text{TRUE}$. 7. Ложь. 8. Истина. 9. Ложь. 10. Ложь. 11. Истина. 12. Ложь. 13. Истина. 14. Ложь. 15. Истина. 16. 10. 17. 6. 18. 8. 19. 2. 20. 1. 21. 8. 22. 8. 23. 6. 24. $A = 1$, $B = 0$, $C = 0$, $D = 1$. 25. 18. 26. Да. 27. Нет. 28. Нет. 29. Фролов, Самсонов. 30. Мидова, Соколова. 31. Никита — первое, Руслан — четвертое, Сергей — второе, Толя — третье. 32. Максим. 33. Вика. 34. Аня. 35. Учитель музыки — Елизавета Евгеньевна, преподаватель физкультуры — Мария Михайловна, педагог по химии — Лада Львовна, историк — Надежда Николаевна. 36. Зеленый ошейник — овчарка, синий ошейник — ротвейлер, черный ошейник — такса, красный ошейник — бульдог. 37. Хозяйка черной кошки — Маша, хозяйка трехцветной кошки — Ира, хозяйка серой кошки — Света, хозяйка белой кошки — Валя.

Построение таблиц истинности логических выражений

1. 8. 2. 32. 3. Да. 4. Да. 5. Да. 6. Нет. 7. Да. 8. Нет. 9. Да.

Комбинационные и переключательные схемы

1. 0. 2. $A \wedge \bar{B} \vee \bar{A} \wedge B$. 3. 0. 4. $\bar{A} \wedge B \wedge \bar{C} \vee A \wedge \bar{B} \wedge C$. 5. $\bar{B} \wedge C \vee A \wedge C$. 6. A.

Теория игр

1. При правильной игре выиграет первый игрок; для этого первый игрок первым ходом должен получить комбинацию (3,5). 2. Выигрывает второй игрок — своим первым ходом он должен получить одну из ситуаций: (5,6) или (4,6) или (7,3). 3. Выигрывает первый игрок — своим первым ходом он должен получить ситуацию (2,4,6).

Глава 3

Моделирование

1. 09:00. 2. 10:55. 3. Да. 4. Нет. 5. Да. 6. 3.5 часа. 7. 3.5 часа.

8. Матрица смежности:

| | | | | |
|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 |

Матрица инцидентности:

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 0 | 0 | 1 |

Список пар вершин:
 $(\{1, 2\}, \{1, 3\}, \{1, 4\}, \{3, 4\}, \{4, 5\})$.

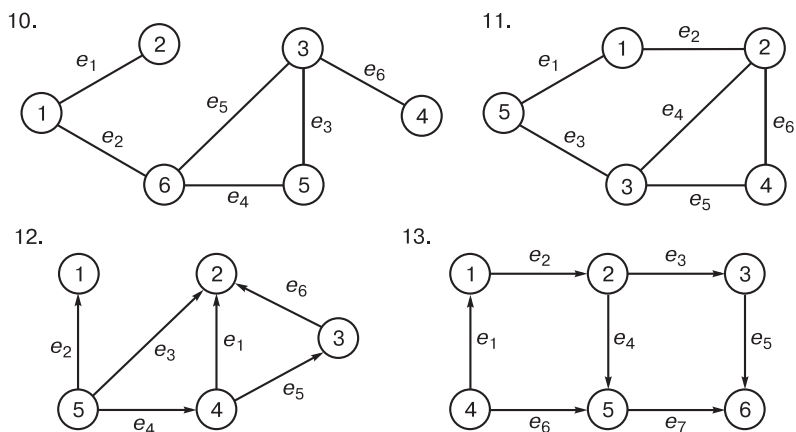
9. Матрица смежности:

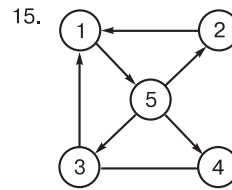
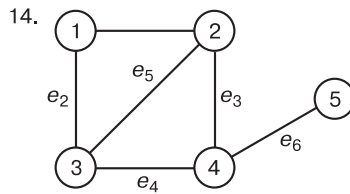
| | | | | |
|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 |

Матрица инцидентности:

| | | | | |
|----|----|----|----|----|
| -1 | -1 | -1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | -1 | 0 |
| 0 | 0 | 1 | 1 | -1 |
| 0 | 0 | 0 | 0 | 1 |

Список пар вершин:
 $((1, 2), (1, 3), (1, 4), (3, 4), (4, 5))$.





16. 0.5. 17. -8.08. 18. 6. 19. 5. 20. 5. 21. 1. 22. 1.5127.
 23. 1.4233. 24. 2.5792. 25. -1.1347. 26. 2.0867. 27. -0.4353.
 28. 0.2210. 29. -2.6129. 30. $x_2 = 1$, $x_4 = 2$, $F(X) = 1$. 31. $x_6 = 2/3$,
 $x_2 = 2/3$, $x_1 = 5/3$, $F(X) = 15$. 32. $x_2 = 1.25$, $x_1 = 1.5$, $F(X) = 11.25$.
 33. $x_1 = 1$, $x_4 = 0$, $F(X) = 1$. 34. $x_1 = 3$, $x_5 = 2$, $x_3 = 6$, $F(X) = -3$. 35. $x_1 = 1$,
 $x_4 = 0$, $F(X) = 1$. 36. (1,4), (4,3), (3,5), (5,2), (2,1), длина маршрута
 равна = 14. 37. (1,4), (4,3), (3,2), (2,5), (5,1), длина маршрута равна
 10. 38. (3,1), (1,4), (4,2), (2,3), длина маршрута равна 23. (4,2), (2,3),
 (3,1), (1,4), длина маршрута равна 14. 40. (1,4), (4,2), (2,3), (3,1),
 длина маршрута равна 12. 41. (2,4), (4,3), (3,1), (1,2), длина маршрута
 равна 18. 42. $x = 2$, $y = 1$, $F = 7$. 43. $x = 6$, $y = 0$, $F = 12$. 44. $x = 3$,
 $y = 2$, $F = 8$. 45. $x = 3.3333$, $y = 1.333$, $F = 12.67$.

Принципы организации и функционирования вычислительных сетей

1. ecafgbd. 2. ecafgbd. 3. dcab. 4. bfc dij. 5. dacb. 6. ГВВА. 7. ВБАГ.
 8. Нет. 9. Да. 10. 670. 11. 450. 12. 616.

Обработка информации в электронных таблицах

1. 23. 2. 14. 3. = \$C\$1*\$B3*\$C3. 4. =\$B\$1*\$D\$1*\$B3. 5. 11. 6.
 3. 7. 2. 8. 8. 9. 1. 10. 23. 11. 23. 12. Да. 13. Нет. 14. Да. 15. Нет. 16. D.
 17. D. 18. 1. 19. Да. 20. 3. 21. 6. 22. 2.

Организация баз данных

1. 3. 2. 1. 3. 3. 4. 2. 5. 2. 6. 2. 7. 1, 2, 4, 6. 8. 5. 9. 29. 10. 11. 11. 14.

Организация файловых систем

1. Да. 2. E:\Школа\Физика\Задание3. 3. A:\TOM3\Дос3.
 4. D:\SCHOOL\DOCUMENTS\TASKS. 5. гдавбе. 6. Нет.

Глава 4

Алгоритм. Исполнитель алгоритма

1. СААВАА. 2. 191. 3. CDEFАА. 4. 255. 5. 254. 6. 14. 7. 10. 8. 136. 9. 1, 7, 9. 10. 2, 1, 3. 11. Да. 12. Нет. 13. Да. 14. Да. 15. Да. 16. Три квадрата. 17. 72. 18. 4144. 19. А6. 20. А6, Е7. 21. 12212. 22. 21221. 23. 36. 24. 2. 25. В. 26. $\overline{C} \wedge A$. 27. 1 2. 28. Электрат.

Глава 5

Оператор присваивания, ввод-вывод данных

1. 8. 2. -5. 3. $x = 2, y = 15, t = 15$. 4. $x = 2, y = -5, t = -15$. 5. $a = 0, b = 1000$. 6. 5. 7. 4. 8. $a = 0, b = 0$. 9. true.

Структуры: циклическая, ветвление

1. 16. 2. 28. 3. 45. 4. 8. 5. 1000. 6. 64. 7. 4. 8. Да. 9. Да. 10. Нет. 12. 3. 13. 12. 14. 8. 15. 22. 16. 2.

Работа с массивами

1. 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0. 2. 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10. 3. 8. 4. 48. 5. 64. 6. 4. 7. 3. 8. 7. 9. Да. 10. Две строки в таблице. 15. -0.5. 16. -2. 17. 3. 18. 2. 19. 17. 20. -17.

Литература

1. Демонстрационный вариант ЕГЭ, 2004 г.
2. Демонстрационный вариант ЕГЭ, 2005 г.
3. Демонстрационный вариант ЕГЭ, 2006 г.
4. Демонстрационный вариант ЕГЭ, 2007 г.
5. Демонстрационный вариант ЕГЭ, 2008 г.
6. Демонстрационный вариант ЕГЭ, 2009 г.
7. Демонстрационный вариант ЕГЭ, 2010 г.

Оглавление

| | |
|--|-----------|
| Предисловие | 3 |
| Глава 1. Информационные процессы и системы | 5 |
| 1.1. Информация и ее кодирование | 5 |
| 1.1.1. Понятие информации | 5 |
| 1.1.2. Методы измерения количества информации | 5 |
| 1.2. Процесс передачи информации | 12 |
| 1.3. Представление числовой информации | 15 |
| 1.3.1. Перевод чисел между системами счисления | 16 |
| 1.3.2. Быстрый перевод чисел между системами счисления | 20 |
| 1.3.3. Представление чисел в прямом, обратном и дополнительных кодах | 36 |
| 1.3.4. Модифицированные обратный и дополнительный коды | 39 |
| 1.3.5. Представление чисел с фиксированной и плавающей запятой | 41 |
| 1.3.6. Арифметические операции над числами | 44 |
| 1.4. Кодирование текстовой информации | 49 |
| 1.5. Кодирование графической информации | 52 |
| 1.6. Кодирование звуковой информации | 59 |
| 1.7. Элементы теории множеств | 62 |
| 1.8. Комбинаторика | 66 |
| Задачи для самостоятельной работы | 70 |
| Глава 2. Алгебра логики | 86 |
| 2.1. Основные понятия алгебры логики | 86 |
| 2.2. Логические выражения и их преобразование | 87 |
| 2.3. Построение таблиц истинности логических выражений | 109 |
| 2.4. Представление логических формул в нормальной форме | 119 |

| | |
|--|------------|
| 2.5. Логические схемы | 122 |
| 2.5.1. Переключательная схема | 122 |
| 2.5.2. Комбинационная схема. | 124 |
| 2.6. Теория игр. | 128 |
| Задачи для самостоятельного решения | 144 |
| Глава 3. Средства ИКТ | 153 |
| 3.1. Обработка информации в электронных таблицах . . . | 153 |
| 3.1.1. Автозаполнение. | 155 |
| 3.1.2. Относительная и абсолютная адресация | 156 |
| 3.1.3. Функции электронных таблиц | 157 |
| 3.2. Организация баз данных | 165 |
| 3.3. Моделирование | 174 |
| 3.3.1. Приближенное решение уравнений | 196 |
| 3.3.2. Поиски экстремумов функций | 200 |
| 3.4. Принципы организации и функционирования
вычислительных сетей | 202 |
| 3.5. Организация файловых систем | 212 |
| Задачи для самостоятельного решения | 216 |
| Глава 4. Алгоритм. Исполнитель алгоритма | 240 |
| Задачи для самостоятельного выполнения | 257 |
| Глава 5. Введение в программирование на языке Паскаль | 267 |
| 5.1. Основные понятия языка Паскаль | 267 |
| 5.2. Типы данных языка Паскаль. | 269 |
| 5.3. Операторы языка Паскаль | 272 |
| 5.3.1. Основные операторы языка Паскаль | 272 |
| 5.3.2. Операторы ввода данных | 273 |
| 5.3.3. Операторы вывода данных | 274 |
| 5.4. Перечисляемый тип | 276 |
| 5.5. Интервальный тип | 278 |
| 5.6. Условный оператор. | 279 |
| 5.7. Оператор выбора | 291 |
| 5.8. Операторы организации циклов | 293 |
| 5.8.1. Оператор цикла for. | 293 |
| 5.8.2. Оператор цикла while. | 295 |
| 5.8.3. Оператор цикла repeat. | 298 |
| 5.8.4. Вложенные операторы циклов | 299 |
| 5.8.5. Операторы break и continue. | 300 |
| 5.9. Работа с массивами. | 305 |
| 5.9.1. Определение массива. | 305 |

| | |
|---|------------|
| 5.9.2. Инициализация массива | 307 |
| 5.9.3. Вывод массива | 311 |
| 5.9.4. Сортировка одномерного массива | 312 |
| 5.9.5. Алгоритмы обработки массивов | 323 |
| 5.9.6. Генерация последовательностей
псевдослучайных чисел | 328 |
| 5.10. Символьные типы данных | 339 |
| 5.10.1. Тип <code>char</code> | 339 |
| 5.10.2. Тип <code>string</code> | 342 |
| 5.11. Записи | 348 |
| 5.11.1. Массив записей | 350 |
| 5.11.2. Инициализация записи. | 351 |
| 5.11.3. Инициализация массива записей. | 352 |
| 5.11.4. Оператор <code>with</code> | 354 |
| 5.11.5. Тестирование программ | 364 |
| 5.12. Множества. | 372 |
| 5.13. Работа с файлами | 374 |
| 5.13.1. Процедуры чтения и записи информации
в файл | 376 |
| 5.13.2. Дополнительные процедуры и функции
для работы с файлами. | 378 |
| 5.13.3. Процедуры для работы с каталогами | 389 |
| 5.14. Подпрограммы | 389 |
| 5.14.1. Описание и вызов подпрограмм. | 390 |
| 5.14.2. Параметры подпрограмм | 391 |
| 5.14.3. Рекурсия | 393 |
| 5.14.4. Локальные и глобальные идентификаторы | 395 |
| 5.14.5. Передача в подпрограмму массивов и строк | 397 |
| Задачи для самостоятельного решения | 400 |
| Глава 6. Справочная информация | 421 |
| 6.1. Действия со степенями. | 421 |
| 6.2. Логарифм числа и его свойства | 421 |
| 6.3. Таблицы сложения и умножения | 421 |
| 6.4. Функции электронных таблиц | 423 |
| 6.5. Коды ASCII | 433 |
| Ответы | 437 |
| Литература | 443 |